Introduction

This document is addressed to application developers. It provides complete information on how to use the STM32WB55xx and STM32WB35xx microcontrollers.

These multiprotocol wireless and ultra-low-power devices embed a powerful ultra-low-power radio compliant with the Bluetooth® Low Energy SIG specification 5.4 and with IEEE 802.15.4-2011. They contain a dedicated Arm® Cortex®-M0+ for performing the real-time low layer operation.

The STM32WB55xx and STM32WB35xx microcontrollers feature different memory sizes, packages and peripherals, and include ST state of the art patented technology.

Related documents

Available from STMicroelectronics web site www.st.com:

STM32WB55xx and STM32WB35xx datasheets
STM32WB55xx and STM32WB35xx errata sheets

For information on the Arm® Cortex®-M4 and Cortex®-M0+ cores, refer, respectively, to the corresponding Technical Reference Manuals, available from the www.arm.com website.

For information on 802.15.4 refer to the IEEE website (www.ieee.org).

For information on Bluetooth® refer to www.bluetooth.com.
## Contents

### 1 Documentation conventions
- 1.1 General information ............................................. 61
- 1.2 List of abbreviations for registers ................................. 61
- 1.3 Glossary .................................................................. 62
- 1.4 Availability of peripherals .......................................... 62

### 2 System and memory overview ........................................... 63
- 2.1 System architecture .................................................. 63
  - 2.1.1 S0: CPU1 (Cortex®-M4) I-bus ................................. 64
  - 2.1.2 S1: CPU1 (Cortex®-M4) D-bus ................................. 64
  - 2.1.3 S2: CPU1 (Cortex®-M4) S-bus ................................. 64
  - 2.1.4 S3: CPU2 (Cortex®-M0+) S-bus ............................... 65
  - 2.1.5 S4, S5: DMA-bus ................................................ 65
  - 2.1.6 S6: Radio system-bus ............................................ 65
  - 2.1.7 BusMatrix ........................................................ 65
- 2.2 Memory organization ................................................ 66
  - 2.2.1 Introduction ....................................................... 66
  - 2.2.2 Memory map and register boundary addresses ............. 67
  - 2.2.3 Bit banding ....................................................... 72
- 2.3 Boot configuration ................................................... 73
- 2.4 CPU2 boot ............................................................. 74
- 2.5 CPU2 SRAM fetch disable .......................................... 75

### 3 Embedded flash memory (FLASH) .......................... 76
- 3.1 Introduction ......................................................... 76
- 3.2 FLASH main features ............................................... 76
- 3.3 FLASH functional description ..................................... 77
  - 3.3.1 Flash memory organization ................................... 77
  - 3.3.2 Empty check ..................................................... 78
  - 3.3.3 Error code correction (ECC) ................................ 78
  - 3.3.4 Read access latency ............................................ 78
  - 3.3.5 Adaptive real-time memory accelerator (ART Accelerator) .......................................................... 80
  - 3.3.6 Flash memory program and erase operations .................. 83
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.7</td>
<td>Flash main memory erase sequences</td>
<td>84</td>
</tr>
<tr>
<td>3.3.8</td>
<td>Flash main memory programming sequences</td>
<td>86</td>
</tr>
<tr>
<td>3.4</td>
<td>FLASH option bytes</td>
<td>91</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Option bytes description</td>
<td>91</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Option bytes programming</td>
<td>98</td>
</tr>
<tr>
<td>3.5</td>
<td>FLASH UID64</td>
<td>101</td>
</tr>
<tr>
<td>3.6</td>
<td>Flash memory protection</td>
<td>102</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Read protection (RDP)</td>
<td>102</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Proprietary code readout protection (PCROP)</td>
<td>106</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Write protection (WRP)</td>
<td>107</td>
</tr>
<tr>
<td>3.6.4</td>
<td>CPU2 security (ESE)</td>
<td>108</td>
</tr>
<tr>
<td>3.7</td>
<td>FLASH program/erase suspension</td>
<td>109</td>
</tr>
<tr>
<td>3.8</td>
<td>FLASH interrupts</td>
<td>110</td>
</tr>
<tr>
<td>3.9</td>
<td>Register access protection</td>
<td>110</td>
</tr>
<tr>
<td>3.10</td>
<td>FLASH registers</td>
<td>111</td>
</tr>
<tr>
<td>3.10.1</td>
<td>Flash memory access control register (FLASH_ACR)</td>
<td>111</td>
</tr>
<tr>
<td>3.10.2</td>
<td>Flash memory key register (FLASH_KEYR)</td>
<td>112</td>
</tr>
<tr>
<td>3.10.3</td>
<td>Flash memory option key register (FLASH_OPTKEYR)</td>
<td>112</td>
</tr>
<tr>
<td>3.10.4</td>
<td>Flash memory status register (FLASH_SR)</td>
<td>113</td>
</tr>
<tr>
<td>3.10.5</td>
<td>Flash memory control register (FLASH_CR)</td>
<td>114</td>
</tr>
<tr>
<td>3.10.6</td>
<td>Flash memory ECC register (FLASH_ECCR)</td>
<td>116</td>
</tr>
<tr>
<td>3.10.7</td>
<td>Flash memory option register (FLASH_OPTR)</td>
<td>117</td>
</tr>
<tr>
<td>3.10.8</td>
<td>Flash memory PCROP zone A start address register (FLASH_PCROP1ASR)</td>
<td>119</td>
</tr>
<tr>
<td>3.10.9</td>
<td>Flash memory PCROP zone A end address register (FLASH_PCROP1AER)</td>
<td>120</td>
</tr>
<tr>
<td>3.10.10</td>
<td>Flash memory WRP area A address register (FLASH_WRP1AR)</td>
<td>120</td>
</tr>
<tr>
<td>3.10.11</td>
<td>Flash memory WRP area B address register (FLASH_WRP1BR)</td>
<td>121</td>
</tr>
<tr>
<td>3.10.12</td>
<td>Flash memory PCROP zone B start address register (FLASH_PCROP1BSR)</td>
<td>121</td>
</tr>
<tr>
<td>3.10.13</td>
<td>Flash memory PCROP zone B end address register (FLASH_PCROP1BER)</td>
<td>122</td>
</tr>
<tr>
<td>3.10.14</td>
<td>Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR)</td>
<td>122</td>
</tr>
<tr>
<td>3.10.15</td>
<td>Flash memory CPU2 access control register (FLASH_C2ACR)</td>
<td>122</td>
</tr>
<tr>
<td>3.10.16</td>
<td>Flash memory CPU2 status register (FLASH_C2SR)</td>
<td>123</td>
</tr>
<tr>
<td>3.10.17</td>
<td>Flash memory CPU2 control register (FLASH_C2CR)</td>
<td>125</td>
</tr>
<tr>
<td>3.10.18</td>
<td>Secure flash memory start address register (FLASH_SFR)</td>
<td>126</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10.19</td>
<td>Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)</td>
<td>127</td>
</tr>
<tr>
<td>3.10.20</td>
<td>FLASH register map</td>
<td>129</td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>Radio system</td>
<td>131</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>131</td>
</tr>
<tr>
<td>4.2</td>
<td>Main features</td>
<td>131</td>
</tr>
<tr>
<td>4.3</td>
<td>Radio system functional description</td>
<td>132</td>
</tr>
<tr>
<td>4.3.1</td>
<td>General description</td>
<td>132</td>
</tr>
<tr>
<td><strong>5</strong></td>
<td>Cyclic redundancy check calculation unit (CRC)</td>
<td>133</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>133</td>
</tr>
<tr>
<td>5.2</td>
<td>CRC main features</td>
<td>133</td>
</tr>
<tr>
<td>5.3</td>
<td>CRC functional description</td>
<td>134</td>
</tr>
<tr>
<td>5.3.1</td>
<td>CRC block diagram</td>
<td>134</td>
</tr>
<tr>
<td>5.3.2</td>
<td>CRC internal signals</td>
<td>134</td>
</tr>
<tr>
<td>5.3.3</td>
<td>CRC operation</td>
<td>134</td>
</tr>
<tr>
<td>5.4</td>
<td>CRC registers</td>
<td>136</td>
</tr>
<tr>
<td>5.4.1</td>
<td>CRC data register (CRC_DR)</td>
<td>136</td>
</tr>
<tr>
<td>5.4.2</td>
<td>CRC independent data register (CRC_IDR)</td>
<td>136</td>
</tr>
<tr>
<td>5.4.3</td>
<td>CRC control register (CRC_CR)</td>
<td>137</td>
</tr>
<tr>
<td>5.4.4</td>
<td>CRC initial value (CRC_INIT)</td>
<td>138</td>
</tr>
<tr>
<td>5.4.5</td>
<td>CRC polynomial (CRC_POL)</td>
<td>138</td>
</tr>
<tr>
<td>5.4.6</td>
<td>CRC register map</td>
<td>139</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td>Power control (PWR)</td>
<td>140</td>
</tr>
<tr>
<td>6.1</td>
<td>Power supplies</td>
<td>140</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Independent analog peripherals supply</td>
<td>144</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Independent USB transceivers supply</td>
<td>144</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Independent LCD supply (available only on STM32WB55xx)</td>
<td>144</td>
</tr>
<tr>
<td>6.1.4</td>
<td>Battery backup domain</td>
<td>145</td>
</tr>
<tr>
<td>6.1.5</td>
<td>Voltage regulator</td>
<td>146</td>
</tr>
<tr>
<td>6.1.6</td>
<td>Dynamic voltage scaling management</td>
<td>147</td>
</tr>
<tr>
<td>6.2</td>
<td>Power supply supervisor</td>
<td>147</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)</td>
<td>147</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Programmable voltage detector (PVD)</td>
<td>149</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Peripheral voltage monitoring (PVM)</td>
<td>150</td>
</tr>
<tr>
<td>6.3</td>
<td>CPU2 boot</td>
<td>151</td>
</tr>
<tr>
<td>6.4</td>
<td>Low-power modes</td>
<td>153</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Run mode</td>
<td>159</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Low-power run mode (LP run)</td>
<td>159</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Entering Low-power mode</td>
<td>160</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Exiting Low-power mode</td>
<td>160</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Sleep mode</td>
<td>162</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Low-power sleep mode (LP sleep)</td>
<td>163</td>
</tr>
<tr>
<td>6.4.7</td>
<td>Stop0 mode</td>
<td>164</td>
</tr>
<tr>
<td>6.4.8</td>
<td>Stop1 mode</td>
<td>166</td>
</tr>
<tr>
<td>6.4.9</td>
<td>Stop2 mode</td>
<td>167</td>
</tr>
<tr>
<td>6.4.10</td>
<td>Standby mode</td>
<td>169</td>
</tr>
<tr>
<td>6.4.11</td>
<td>Shutdown mode</td>
<td>171</td>
</tr>
<tr>
<td>6.4.12</td>
<td>Auto wake-up from Low-power mode</td>
<td>172</td>
</tr>
<tr>
<td>6.5</td>
<td>Real-time radio information</td>
<td>173</td>
</tr>
<tr>
<td>6.6</td>
<td>PWR registers</td>
<td>174</td>
</tr>
<tr>
<td>6.6.1</td>
<td>PWR control register 1 (PWR_CR1)</td>
<td>174</td>
</tr>
<tr>
<td>6.6.2</td>
<td>PWR control register 2 (PWR_CR2)</td>
<td>175</td>
</tr>
<tr>
<td>6.6.3</td>
<td>PWR control register 3 (PWR_CR3)</td>
<td>176</td>
</tr>
<tr>
<td>6.6.4</td>
<td>PWR control register 4 (PWR_CR4)</td>
<td>178</td>
</tr>
<tr>
<td>6.6.5</td>
<td>PWR status register 1 (PWR_SR1)</td>
<td>179</td>
</tr>
<tr>
<td>6.6.6</td>
<td>PWR status register 2 (PWR_SR2)</td>
<td>180</td>
</tr>
<tr>
<td>6.6.7</td>
<td>PWR status clear register (PWR_SCR)</td>
<td>181</td>
</tr>
<tr>
<td>6.6.8</td>
<td>PWR control register 5 (PWR_CR5)</td>
<td>182</td>
</tr>
<tr>
<td>6.6.9</td>
<td>PWR Port A pull-up control register (PWR_PUCRA)</td>
<td>183</td>
</tr>
<tr>
<td>6.6.10</td>
<td>PWR Port A pull-down control register (PWR_PDCRA)</td>
<td>184</td>
</tr>
<tr>
<td>6.6.11</td>
<td>PWR Port B pull-up control register (PWR_PUCRB)</td>
<td>184</td>
</tr>
<tr>
<td>6.6.12</td>
<td>PWR Port B pull-down control register (PWR_PDCRB)</td>
<td>185</td>
</tr>
<tr>
<td>6.6.13</td>
<td>PWR Port C pull-up control register (PWR_PUCRC)</td>
<td>185</td>
</tr>
<tr>
<td>6.6.14</td>
<td>PWR Port C pull-down control register (PWR_PDCRC)</td>
<td>186</td>
</tr>
<tr>
<td>6.6.15</td>
<td>PWR Port D pull-up control register (PWR_PUCRD) (STM32WB55xx only)</td>
<td>186</td>
</tr>
<tr>
<td>6.6.16</td>
<td>PWR Port D pull-down control register (PWR_PDCRD) (STM32WB55xx only)</td>
<td>187</td>
</tr>
<tr>
<td>6.6.17</td>
<td>PWR Port E pull-up control register (PWR_PUCRE)</td>
<td>187</td>
</tr>
<tr>
<td>6.6.18</td>
<td>PWR Port E pull-down control register (PWR_PDCRE)</td>
<td>188</td>
</tr>
</tbody>
</table>
7 Peripherals interconnect matrix .......................................................... 196
   7.1 Introduction ............................................................................. 196
   7.2 Interconnect matrix implementation ........................................... 196
   7.3 Connection summary ................................................................. 196
   7.4 Interconnection details ............................................................... 197
      7.4.1 From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2) .......... 197
      7.4.2 From timer (TIM1/TIM2) and EXTI to ADC (ADC1) .......... 198
      7.4.3 From ADC (ADC1) to timer (TIM1) .................................. 198
      7.4.4 From HSE, LSE, LSI, MSI, MCO, RTC to timers
               (TIM2/TIM16/TIM17) .................................................. 198
      7.4.5 From RTC, COMP1, COMP2 to low-power timers (LPTIM1/LPTIM2). 199
      7.4.6 From timer (TIM1/TIM2) to comparators (COMP1/COMP2) ..... 199
      7.4.7 From USB to timer (TIM2) .............................................. 200
      7.4.8 From internal analog to ADC1 ........................................ 200
      7.4.9 From comparators (COMP1/COMP2) to timers
               (TIM1/TIM2/TIM16/TIM17) ...................................... 200
      7.4.10 From system errors to timers (TIM1/TIM16/TIM17) ............ 201
      7.4.11 From timers (TIM16/TIM17) to IRTIM ............................ 201

8 Reset and clock control (RCC) .......................................................... 202
   8.1 Reset .................................................................................... 202
      8.1.1 Power reset ................................................................. 202
      8.1.2 System reset ............................................................... 202
      8.1.3 Backup domain reset ..................................................... 204
   8.2 Clocks ............................................................................... 204
      8.2.1 HSE clock ................................................................. 208
      8.2.2 HSI16 clock ............................................................... 209
      8.2.3 MSI clock ................................................................. 209
      8.2.4 HSI48 clock ............................................................... 210
      8.2.5 PLLs ....................................................................... 211
8.2.6 LSE clock ...................................................... 211
8.2.7 LSI1 clock ...................................................... 212
8.2.8 LSI2 clock ...................................................... 213
8.2.9 System clock (SYSCLK) selection ...................................................... 213
8.2.10 Clock source frequency versus voltage scaling ...................................................... 213
8.2.11 Clock security system (CSS) on HSE ...................................................... 214
8.2.12 Clock security system on LSE (LSECSS) ...................................................... 214
8.2.13 LSI source selection ...................................................... 215
8.2.14 SMPS step-down converter clock ...................................................... 215
8.2.15 ADC clock ...................................................... 216
8.2.16 RTC clock ...................................................... 216
8.2.17 Timer clock ...................................................... 216
8.2.18 Watchdog clock ...................................................... 217
8.2.19 True RNG clock ...................................................... 217
8.2.20 Clock-out capability ...................................................... 217
8.2.21 Internal/external clock measurement with TIM16/TIM17 ...................................................... 218
8.2.22 Peripheral clocks enable ...................................................... 219
8.3 Low-power modes ...................................................... 221
8.4 RCC registers ...................................................... 223
  8.4.1 RCC clock control register (RCC_CR) ...................................................... 223
  8.4.2 RCC internal clock sources calibration register (RCC_ICSCR) ...................... 226
  8.4.3 RCC clock configuration register (RCC_CFGR) ...................................................... 227
  8.4.4 RCC PLL configuration register (RCC_PLLCFGR) ...................................................... 229
  8.4.5 RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR) ...................................................... 233
  8.4.6 RCC clock interrupt enable register (RCC_CIER) ...................................................... 235
  8.4.7 RCC clock interrupt flag register (RCC_CIFR) ...................................................... 236
  8.4.8 RCC clock interrupt clear register (RCC_CICR) ...................................................... 238
  8.4.9 RCC SMPS step-down converter control register (RCC_SMPSCR) ...................................................... 239
  8.4.10 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) ...................................................... 240
  8.4.11 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) ...................................................... 241
  8.4.12 RCC AHB3 and AHB4 peripheral reset register (RCC_AHB3RSTR) ...................................................... 242
  8.4.13 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1) ...................................................... 243
  8.4.14 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2) ...................................................... 245
  8.4.15 RCC APB2 peripheral reset register (RCC_APB2RSTR) ...................................................... 245
  8.4.16 RCC APB3 peripheral reset register (RCC_APB3RSTR) ...................................................... 246
  8.4.17 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR) ...................................................... 247
  8.4.18 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR) ...................................................... 248
8.4.19 RCC AHB3 and AHB4 peripheral clock enable register (RCC_AHB3ENR) ........................................ 249
8.4.20 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1) .................................................. 250
8.4.21 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2) .................................................. 252
8.4.22 RCC APB2 peripheral clock enable register (RCC_APB2ENR) ......................................................... 252
8.4.23 RCC AHB1 peripheral clocks enable in Sleep modes register (RCC_AHB1SMENR) .............................. 253
8.4.24 RCC AHB2 peripheral clocks enable in Sleep modes register (RCC_AHB2SMENR) ............................. 254
8.4.25 RCC AHB3 and AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR) ................................................................................................................. 256
8.4.26 RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC_APB1SMENR1) ......................... 257
8.4.27 RCC APB1 peripheral clocks enable in Sleep mode register 2 (RCC_APB1SMENR2) ......................... 259
8.4.28 RCC APB2 peripheral clocks enable in Sleep mode register (RCC_APB2SMENR) ............................... 259
8.4.29 RCC peripherals independent clock configuration register (RCC_CCIPR) .......................................... 261
8.4.30 RCC backup domain control register (RCC_BDCR) ............................................................................ 262
8.4.31 RCC control/status register (RCC_CSR) .......................................................................................... 264
8.4.32 RCC clock recovery RC register (RCC_CRRCR) ................................................................................. 266
8.4.33 RCC clock HSE register (RCC_HSECR) ............................................................................................ 267
8.4.34 RCC extended clock recovery register (RCC_EXTCFGR) ................................................................. 268
8.4.35 RCC CPU2 AHB1 peripheral clock enable register (RCC_C2AHB1ENR) ............................................. 270
8.4.36 RCC CPU2 AHB2 peripheral clock enable register (RCC_C2AHB2ENR) ............................................. 271
8.4.37 RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC_C2AHB3ENR) ......................... 272
8.4.38 RCC CPU2 APB1 peripheral clock enable register 1 (RCC_C2APB1ENR1) ........................................... 273
8.4.39 RCC CPU2 APB1 peripheral clock enable register 2 (RCC_C2APB1ENR2) ........................................... 275
8.4.40 RCC CPU2 APB2 peripheral clock enable register (RCC_C2APB2ENR) ............................................. 275
8.4.41 RCC CPU2 APB3 peripheral clock enable register (RCC_C2APB3ENR) ............................................. 276
8.4.42 RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC_C2AHB1SMENR) .............. 277
8.4.43 RCC CPU2 AHB2 peripheral clocks enable in Sleep modes register (RCC_C2AHB2SMENR) .............. 278
9 General-purpose I/Os (GPIO) .................................................. 291

9.1 Introduction ................................................................. 291
9.2 GPIO main features ......................................................... 291
9.3 GPIO implementation ....................................................... 291
9.4 GPIO functional description ............................................. 291
  9.4.1 General-purpose I/O (GPIO) .......................................... 294
  9.4.2 I/O pin alternate function multiplexer and mapping .......... 294
  9.4.3 I/O port control registers ........................................... 295
  9.4.4 I/O port data registers .............................................. 295
  9.4.5 I/O data bitwise handling .......................................... 295
  9.4.6 GPIO locking mechanism ............................................ 296
  9.4.7 I/O alternate function input/output ................................ 296
  9.4.8 External interrupt/wakeup lines .................................. 296
  9.4.9 Input configuration .................................................. 296
  9.4.10 Output configuration ............................................... 297
  9.4.11 Alternate function configuration .................................. 298
  9.4.12 Analog configuration ............................................... 298
  9.4.13 Using the LSE oscillator pins as GPIOs ......................... 299
  9.4.14 Using the GPIO pins in the RTC supply domain ............... 299
  9.4.15 Using PH3 as GPIO ................................................. 299

9.5 GPIO registers .............................................................. 300
  9.5.1 GPIO port mode register (GPIOx_MODER) (x = A to E and H) .... 300
  9.5.2 GPIO port output type register (GPIOx_OTYPER) (x = A to E and H) 301
  9.5.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to E and H) 301
9.5.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR)  
(x = A to E and H) .................................................. 302
9.5.5 GPIO port input data register (GPIOx_IDR)  
(x = A to E and H) .................................................. 302
9.5.6 GPIO port output data register (GPIOx_ODR)  
(x = A to E and H) .................................................. 303
9.5.7 GPIO port bit set/reset register (GPIOx_BSRR)  
(x = A to E and H) .................................................. 303
9.5.8 GPIO port configuration lock register (GPIOx_LCKR)  
(x = A to E and H) .................................................. 304
9.5.9 GPIO alternate function low register (GPIOx_AFRL)  
(x = A to E and H) .................................................. 305
9.5.10 GPIO alternate function high register (GPIOx_AFRH)  
(x = A to E and H) .................................................. 306
9.5.11 GPIO port bit reset register (GPIOx_BRR) (x = A to E and H) .... 307
9.5.12 GPIO register map ........................................... 308

10 System configuration controller (SYSCFG) .................. 311
10.1 SYSCFG main features ....................................... 311
10.2 SYSCFG registers ............................................. 311
10.2.1 SYSCFG memory remap register (SYSCFG_MEMRMP) .... 311
10.2.2 SYSCFG configuration register 1 (SYSCFG_CFGR1) .... 312
10.2.3 SYSCFG external interrupt configuration register 1  
(SYSCFG_EXTICR1) ................................................. 313
10.2.4 SYSCFG external interrupt configuration register 2  
(SYSCFG_EXTICR2) ................................................. 314
10.2.5 SYSCFG external interrupt configuration register 3  
(SYSCFG_EXTICR3) ................................................. 315
10.2.6 SYSCFG external interrupt configuration register 4  
(SYSCFG_EXTICR4) ................................................. 317
10.2.7 SYSCFG SRAM2 control and status register (SYSCFG_SCSR) .... 318
10.2.8 SYSCFG configuration register 2 (SYSCFG_CFGR2) .... 319
10.2.9 SYSCFG SRAM2 write protection register (SYSCFG_SWRP1) .... 320
10.2.10 SYSCFG SRAM2 key register (SYSCFG_SKR) ............ 320
10.2.11 SYSCFG SRAM2 write protection register 2 (SYSCFG_SWRP2) .... 320
10.2.12 SYSCFG CPU1 interrupt mask register 1 (SYSCFG_IMR1) .... 321
10.2.13 SYSCFG CPU1 interrupt mask register 2 (SYSCFG_IMR2) .... 321
10.2.14 SYSCFG CPU2 interrupt mask register 1 (SYSCFG_C2IMR1) .... 322
10.2.15 SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2) .... 323
10.2.16 SYSCFG secure IP control register (SYSCFG_SIPCR) .... 323
11 Direct memory access controller (DMA) ........................................ 327

11.1 Introduction ............................................................. 327
11.2 DMA main features .................................................... 327
11.3 DMA implementation .................................................. 328
  11.3.1 DMA1 and DMA2 ................................................ 328
  11.3.2 DMA request mapping ....................................... 329
11.4 DMA functional description .......................................... 329
  11.4.1 DMA block diagram ........................................... 329
  11.4.2 DMA transfers ................................................ 331
  11.4.3 DMA arbitration ............................................... 332
  11.4.4 DMA channels ................................................ 332
  11.4.5 DMA data width, alignment, and endianness ............. 336
  11.4.6 DMA error management ..................................... 337
11.5 DMA interrupts .......................................................... 338
11.6 DMA registers .......................................................... 338
  11.6.1 DMA interrupt status register (DMA_ISR) ............... 338
  11.6.2 DMA interrupt flag clear register (DMA_IFCR) .......... 341
  11.6.3 DMA channel x configuration register (DMA_CCRx) .... 342
  11.6.4 DMA channel x number of data to transfer register (DMA_CNDTRx) 345
  11.6.5 DMA channel x peripheral address register (DMA_CPARx) ... 345
  11.6.6 DMA channel x memory address register (DMA_CMARx) .. 346
  11.6.7 DMA register map .............................................. 346

12 DMA request multiplexer (DMAMUX) ...................................... 349

12.1 Introduction ............................................................ 349
12.2 DMAMUX main features .............................................. 350
12.3 DMAMUX implementation ............................................ 350
  12.3.1 DMAMUX instantiation ...................................... 350
  12.3.2 DMAMUX mapping ............................................. 350
12.4 DMAMUX functional description .................................... 353
  12.4.1 DMAMUX block diagram .................................... 353
  12.4.2 DMAMUX signals .............................................. 354
  12.4.3 DMAMUX channels ............................................ 354
  12.4.4 DMAMUX request line multiplexer ......................... 354
## 12.4.5 DMAMUX request generator ........................................... 357
## 12.5 DMAMUX interrupts .................................................... 358
## 12.6 DMAMUX registers .................................................... 359
  12.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) .................................................. 359
  12.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) ........................................ 360
  12.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) .................................................. 360
  12.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) .................................................. 361
  12.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR) .................................................. 362
  12.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) .................................................. 362
  12.6.7 DMAMUX register map .................................................. 363

## 13 Nested vectored interrupt controller (NVIC) .......................... 365
  13.1 NVIC main features .................................................... 365
  13.2 NVIC implementation .................................................. 365
  13.3 Interrupt block diagram .............................................. 365
  13.4 Interrupt and exception vectors ...................................... 366
  13.5 Interrupt list .......................................................... 372

## 14 Extended interrupt and event controller (EXTI) ....................... 374
  14.1 EXTI main features .................................................... 374
  14.2 EXTI implementation .................................................. 374
  14.3 EXTI block diagram .................................................... 375
    14.3.1 EXTI connections between peripherals and CPU .................. 376
  14.4 EXTI functional description ........................................... 377
    14.4.1 EXTI configurable event input wakeup .......................... 377
    14.4.2 EXTI direct event input wakeup ................................. 379
  14.5 EXTI functional behavior .............................................. 379
  14.6 EXTI registers ........................................................ 381
    14.6.1 EXTI rising trigger selection register (EXTI_RTSR1) .......... 381
    14.6.2 EXTI falling trigger selection register (EXTI_FTSR1) ......... 382
    14.6.3 EXTI software interrupt event register (EXTI_SWIER1) ....... 382
14.6.4 EXTI pending register (EXTI_PR1) ........................................... 383
14.6.5 EXTI rising trigger selection register (EXTI_RTSR2) .................. 383
14.6.6 EXTI falling trigger selection register (EXTI_FTSR2) ................. 384
14.6.7 EXTI software interrupt event register (EXTI_SWIER2) .............. 385
14.6.8 EXTI pending register (EXTI_PR2) ........................................... 386
14.6.9 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1) ....... 386
14.6.10 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1) .. 387
14.6.11 EXTI CPU wakeup with event mask register (EXTI_EMR1) ........... 387
14.6.12 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1) ..... 388
14.6.13 EXTI CPU wakeup with interrupt mask register (EXTI_IMR2) ....... 388
14.6.14 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2) .. 389
14.6.15 EXTI CPU wakeup with event mask register (EXTI_EMR2) .......... 389
14.6.16 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2) ....... 390
14.6.17 EXTI register map ............................................................ 391

15 Quad-SPI interface (QUADSPI) ................................................. 393
15.1 Introduction ............................................................................. 393
15.2 QUADSPI main features ......................................................... 393
15.3 QUADSPI functional description .............................................. 393
15.3.1 QUADSPI block diagram ..................................................... 393
15.3.2 QUADSPI pins .................................................................. 394
15.3.3 QUADSPI command sequence ........................................... 394
15.3.4 QUADSPI signal interface protocol modes ......................... 396
15.3.5 QUADSPI indirect mode ...................................................... 398
15.3.6 QUADSPI automatic status-polling mode ............................ 399
15.3.7 QUADSPI memory-mapped mode ...................................... 400
15.3.8 QUADSPI flash memory configuration ............................... 401
15.3.9 QUADSPI delayed data sampling ....................................... 401
15.3.10 QUADSPI configuration .................................................... 401
15.3.11 QUADSPI use ................................................................. 402
15.3.12 Sending the instruction only once ...................................... 404
15.3.13 QUADSPI error management ........................................... 404
15.3.14 QUADSPI busy bit and abort functionality ....................... 404
15.3.15 NCS behavior ................................................................. 404
15.4 QUADSPI interrupts ............................................................. 406
15.5 QUADSPI registers ............................................................... 407
15.5.1 QUADSPI control register (QUADSPI_CR) ......................... 407
16  Analog-to-digital converter (ADC) .............................................. 419
   16.1  Introduction ................................................................. 419
   16.2  ADC main features ......................................................... 419
   16.3  ADC implementation ...................................................... 420
   16.4  ADC functional description ............................................ 421
       16.4.1  ADC block diagram ................................................. 421
       16.4.2  ADC pins and internal signals .................................. 422
       16.4.3  ADC clocks ............................................................ 423
       16.4.4  ADC1 connectivity .................................................. 425
       16.4.5  Slave AHB interface ............................................... 426
       16.4.6  ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) .............................................. 426
       16.4.7  Single-ended and differential input channels .................... 427
       16.4.8  Calibration (ADCAL, ADCALDIF, ADC_CALFACT) ................ 427
       16.4.9  ADC on-off control (ADEN, ADDIS, ADRDY) ...................... 430
       16.4.10  Constraints when writing the ADC control bits ................. 431
       16.4.11  Channel selection (SQRx, JSQRx) ................................ 432
       16.4.12  Channel-wise programmable sampling time (SMPR1, SMPR2) ..... 433
       16.4.13  Single conversion mode (CONT = 0) ................................ 433
       16.4.14  Continuous conversion mode (CONT = 1) .......................... 434
       16.4.15  Starting conversions (ADSTART, JADSTART) .................... 435
       16.4.16  ADC timing ............................................................ 436
       16.4.17  Stopping an ongoing conversion (ADSTP, JADSTP) ............... 436
16.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) ........................................ 438
16.4.19 Injected channel management ........................................ 440
16.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) ........ 441
16.4.21 Queue of context for injected conversions ...................... 442
16.4.22 Programmable resolution (RES) - Fast conversion mode ...... 450
16.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) 451
16.4.24 End of conversion sequence (EOS, JEOS) ....................... 451
16.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers) ........................................... 452
16.4.26 Data management ..................................................... 454
16.4.27 Dynamic low-power features ....................................... 459
16.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD1HTx, AWD2HTx, AWDx) . . 464
16.4.29 Oversampler ......................................................... 468
16.4.30 Temperature sensor .................................................. 473
16.4.31 VBAT supply monitoring ............................................. 475
16.4.32 Monitoring the internal voltage reference ...................... 475

16.5 ADC in low-power mode ................................................ 477
16.6 ADC interrupts .......................................................... 478
16.7 ADC registers ............................................................ 479
  16.7.1 ADC interrupt and status register (ADC_ISR) ................. 479
  16.7.2 ADC interrupt enable register (ADC_IER) ..................... 481
  16.7.3 ADC control register (ADC_CR) ................................ 483
  16.7.4 ADC configuration register (ADC_CFGR) ...................... 486
  16.7.5 ADC configuration register 2 (ADC_CFGR2) ................. 490
  16.7.6 ADC sample time register 1 (ADC_SMPR1) .................... 491
  16.7.7 ADC sample time register 2 (ADC_SMPR2) .................... 492
  16.7.8 ADC watchdog threshold register 1 (ADC_TR1) ............. 493
  16.7.9 ADC watchdog threshold register 2 (ADC_TR2) ............. 493
  16.7.10 ADC watchdog threshold register 3 (ADC_TR3) ............ 494
  16.7.11 ADC regular sequence register 1 (ADC_SQR1) ............. 495
  16.7.12 ADC regular sequence register 2 (ADC_SQR2) ............. 496
  16.7.13 ADC regular sequence register 3 (ADC_SQR3) ............. 497
  16.7.14 ADC regular sequence register 4 (ADC_SQR4) ............. 498
  16.7.15 ADC regular data register (ADC_DR) ....................... 498
  16.7.16 ADC injected sequence register (ADC_JSQR) .............. 499
  16.7.17 ADC offset y register (ADC_OFRy) .......................... 501
16.7.18 ADC injected channel y data register (ADC_JDRy) ................. 502
16.7.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) ... 502
16.7.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) ... 503
16.7.21 ADC differential mode selection register (ADC_DIFSEL) .......... 503
16.7.22 ADC calibration factors (ADC_CALFACT) ........................ 504

16.8 ADC common registers ............................................. 504
16.8.1 ADC common status register (ADC_CSR) .......................... 504
16.8.2 ADC common control register (ADC_CCR) .......................... 505

16.9 ADC register map .................................................. 507

17 Voltage reference buffer (VREFBUF) ................................ 510
17.1 Introduction ....................................................... 510
17.2 VREFBUF implementation ........................................... 510
17.3 VREFBUF functional description .................................... 510
17.4 VREFBUF trimming .................................................. 511
17.5 VREFBUF registers .................................................. 511
17.5.1 VREFBUF control and status register (VREFBUF_CSR) ........... 511
17.5.2 VREFBUF calibration control register (VREFBUF_CCR) .......... 512
17.5.3 VREFBUF register map ........................................... 512

18 Comparator (COMP) ................................................... 514
18.1 Introduction ....................................................... 514
18.2 COMP main features ............................................... 514
18.3 COMP functional description ....................................... 515
18.3.1 COMP block diagram ............................................ 515
18.3.2 COMP pins and internal signals ................................... 515
18.3.3 COMP reset and clocks ........................................... 517
18.3.4 Comparator LOCK mechanism ..................................... 517
18.3.5 Window comparator .............................................. 517
18.3.6 Hysteresis ...................................................... 518
18.3.7 Comparator output blanking function ............................... 519
18.3.8 COMP power and speed modes ................................... 519
18.4 COMP low-power modes ............................................ 520
18.5 COMP interrupts ................................................... 520
18.6 COMP registers .................................................... 521
18.6.1 Comparator 1 control and status register (COMP1_CSR) ........ 521
18.6.2 Comparator 2 control and status register (COMP2_CSR) ............. 523
18.6.3 COMP register map ................................................. 526

19 Liquid crystal display controller (LCD) .............................. 527
  19.1 LCD introduction ................................................... 527
  19.2 LCD main features .................................................. 527
  19.3 LCD functional description ....................................... 529
    19.3.1 General description .......................................... 529
    19.3.2 Frequency generator .......................................... 530
    19.3.3 Common driver .................................................. 531
    19.3.4 Segment driver ................................................ 534
    19.3.5 Voltage generator and contrast control ...................... 538
    19.3.6 Double-buffer memory ....................................... 541
    19.3.7 COM and SEG multiplexing .................................... 541
    19.3.8 Flowchart ..................................................... 547
  19.4 LCD low-power modes .............................................. 548
  19.5 LCD interrupts .................................................... 548
  19.6 LCD registers ..................................................... 549
    19.6.1 LCD control register (LCD_CR) ............................. 549
    19.6.2 LCD frame control register (LCD_FCR) ..................... 550
    19.6.3 LCD status register (LCD_SR) .............................. 552
    19.6.4 LCD clear register (LCD_CLR) .............................. 554
    19.6.5 LCD display memory (LCD_RAMx) ............................ 554
    19.6.6 LCD display memory (LCD_RAMx) ............................ 555
    19.6.7 LCD display memory (LCD_RAMx) ............................ 555
    19.6.8 LCD register map .............................................. 555

20 Touch sensing controller (TSC) ...................................... 558
  20.1 Introduction ...................................................... 558
  20.2 TSC main features ................................................ 558
  20.3 TSC functional description ...................................... 559
    20.3.1 TSC block diagram ........................................... 559
    20.3.2 Surface charge transfer acquisition overview .............. 559
    20.3.3 Reset and clocks ............................................. 562
    20.3.4 Charge transfer acquisition sequence ....................... 562
    20.3.5 Spread spectrum feature .................................... 564
### Contents

20.3.6 Max count error ................................................. 564
20.3.7 Sampling capacitor I/O and channel I/O mode selection .... 565
20.3.8 Acquisition mode ............................................. 566
20.3.9 I/O hysteresis and analog switch control .................... 566

20.4 TSC low-power modes ............................................. 567

20.5 TSC interrupts .................................................. 567

20.6 TSC registers .................................................... 567
20.6.1 TSC control register (TSC_CR) ............................... 567
20.6.2 TSC interrupt enable register (TSC_IER) ..................... 570
20.6.3 TSC interrupt clear register (TSC_ICR) ..................... 571
20.6.4 TSC interrupt status register (TSC_ISR) .................... 571
20.6.5 TSC I/O hysteresis control register (TSC_IOHCR) .......... 572
20.6.6 TSC I/O analog switch control register (TSC_IOASCR) .... 572
20.6.7 TSC I/O sampling control register (TSC_IOSCR) .......... 573
20.6.8 TSC I/O channel control register (TSC_IOCCR) .......... 573
20.6.9 TSC I/O group control status register (TSC_IOGCSR) ..... 574
20.6.10 TSC I/O group x counter register (TSC_IOGxCR) ....... 574
20.6.11 TSC register map ............................................ 575

21 True random number generator (RNG) .......................... 577

21.1 Introduction .................................................. 577

21.2 RNG main features .............................................. 577

21.3 RNG functional description .................................... 578
21.3.1 RNG block diagram .......................................... 578
21.3.2 RNG internal signals ........................................ 578
21.3.3 Random number generation .................................. 579
21.3.4 RNG initialization ............................................ 581
21.3.5 RNG operation ............................................... 582
21.3.6 RNG clocking ............................................... 583
21.3.7 Error management ............................................ 583
21.3.8 RNG low-power use .......................................... 584

21.4 RNG interrupts ................................................ 585

21.5 RNG processing time ........................................... 585

21.6 RNG entropy source validation ................................. 585
21.6.1 Introduction ................................................ 585
21.6.2 Validation conditions ................................................. 585
21.7 RNG registers ................................................................. 586
   21.7.1 RNG control register (RNG_CR) ............................... 586
   21.7.2 RNG status register (RNG_SR) ................................... 586
   21.7.3 RNG data register (RNG_DR) .................................... 587
   21.7.4 RNG register map .................................................. 588

22 AES hardware accelerator (AES) ....................................... 589
   22.1 Introduction ............................................................. 589
   22.2 AES main features ..................................................... 589
   22.3 AES implementation ................................................... 589
   22.4 AES functional description .......................................... 590
      22.4.1 AES block diagram .............................................. 590
      22.4.2 AES internal signals ........................................... 590
      22.4.3 AES cryptographic core ......................................... 590
      22.4.4 AES procedure to perform a cipher operation .......... 596
      22.4.5 AES decryption round key preparation ................... 599
      22.4.6 AES ciphertext stealing and data padding ............. 600
      22.4.7 AES task suspend and resume ................................. 600
      22.4.8 AES basic chaining modes (ECB, CBC) .................... 601
      22.4.9 AES counter (CTR) mode ..................................... 606
      22.4.10 AES Galois/counter mode (GCM) ............................ 608
      22.4.11 AES Galois message authentication code (GMAC) .... 613
      22.4.12 AES counter with CBC-MAC (CCM) ......................... 615
      22.4.13 AES data registers and data swapping ................... 620
      22.4.14 AES key registers ............................................ 622
      22.4.15 AES initialization vector registers ........................ 622
      22.4.16 AES DMA interface ........................................... 623
      22.4.17 AES error management ....................................... 624
   22.5 AES interrupts .......................................................... 625
   22.6 AES processing latency .............................................. 625
   22.7 AES registers .......................................................... 626
      22.7.1 AES control register (AES_CR) .............................. 626
      22.7.2 AES status register (AES_SR) ............................... 629
      22.7.3 AES data input register (AES_DINR) ...................... 630
      22.7.4 AES data output register (AES_DOUTR) ................... 630
22.7.5 AES key register 0 (AES_KEYR0) ........................................ 631
22.7.6 AES key register 1 (AES_KEYR1) .................................... 632
22.7.7 AES key register 2 (AES_KEYR2) .................................... 632
22.7.8 AES key register 3 (AES_KEYR3) .................................... 632
22.7.9 AES initialization vector register 0 (AES_IVR0) .................. 633
22.7.10 AES initialization vector register 1 (AES_IVR1) ............... 633
22.7.11 AES initialization vector register 2 (AES_IVR2) ............... 633
22.7.12 AES initialization vector register 3 (AES_IVR3) ............... 634
22.7.13 AES key register 4 (AES_KEYR4) .................................. 634
22.7.14 AES key register 5 (AES_KEYR5) .................................. 634
22.7.15 AES key register 6 (AES_KEYR6) .................................. 635
22.7.16 AES key register 7 (AES_KEYR7) .................................. 635
22.7.17 AES suspend registers (AES_SUSPxR) ......................... 635
22.7.18 AES register map .................................................. 636

23 Public key accelerator (PKA) .............................................. 638
   23.1 Introduction .................................................. 638
   23.2 PKA main features ............................................. 638
   23.3 PKA functional description ..................................... 638
       23.3.1 PKA block diagram ....................................... 638
       23.3.2 PKA internal signals ...................................... 639
       23.3.3 PKA reset and clocks ..................................... 639
       23.3.4 PKA public key acceleration ............................. 639
       23.3.5 Typical applications for PKA ............................ 641
       23.3.6 PKA procedure to perform an operation ................. 643
       23.3.7 PKA error management ..................................... 644
   23.4 PKA operating modes ............................................ 644
       23.4.1 Introduction .............................................. 644
       23.4.2 Montgomery parameter computation ..................... 645
       23.4.3 Modular addition ........................................... 646
       23.4.4 Modular subtraction ....................................... 646
       23.4.5 Modular and Montgomery multiplication ............... 646
       23.4.6 Modular exponentiation ................................... 647
       23.4.7 Modular inversion ......................................... 648
       23.4.8 Modular reduction ......................................... 649
       23.4.9 Arithmetic addition ....................................... 649
       23.4.10 Arithmetic subtraction ................................... 649
### 23.4.1 Arithmetic multiplication

23.4.12 Arithmetic comparison

23.4.13 RSA CRT exponentiation

23.4.14 Point on elliptic curve $F_p$ check

23.4.15 ECC $F_p$ scalar multiplication

23.4.16 ECDSA sign

23.4.17 ECDSA verification

### 23.5 Example of configurations and processing times

23.5.1 Supported elliptic curves

23.5.2 Computation times

### 23.6 PKA interrupts

### 23.7 PKA registers

23.7.1 PKA control register (PKA_CR)

23.7.2 PKA status register (PKA_SR)

23.7.3 PKA clear flag register (PKA_CLRFR)

23.7.4 PKA RAM

23.7.5 PKA register map

### 24 Advanced-control timer (TIM1)

24.1 TIM1 introduction

24.2 TIM1 main features

24.3 TIM1 functional description

24.3.1 Time-base unit

24.3.2 Counter modes

24.3.3 Repetition counter

24.3.4 External trigger input

24.3.5 Clock selection

24.3.6 Capture/compare channels

24.3.7 Input capture mode

24.3.8 PWM input mode

24.3.9 Forced output mode

24.3.10 Output compare mode

24.3.11 PWM mode

24.3.12 Asymmetric PWM mode

24.3.13 Combined PWM mode

24.3.14 Combined 3-phase PWM mode
### 24.3.15 Complementary outputs and dead-time insertion
- Page 699

### 24.3.16 Using the break function
- Page 701

### 24.3.17 Bidirectional break inputs
- Page 707

### 24.3.18 Clearing the OCxREF signal on an external event
- Page 709

### 24.3.19 6-step PWM generation
- Page 710

### 24.3.20 One-pulse mode
- Page 711

### 24.3.21 Retriggerable one pulse mode
- Page 712

### 24.3.22 Encoder interface mode
- Page 713

### 24.3.23 UIF bit remapping
- Page 715

### 24.3.24 Timer input XOR function
- Page 716

### 24.3.25 Interfacing with Hall sensors
- Page 716

### 24.3.26 Timer synchronization
- Page 719

### 24.3.27 ADC synchronization
- Page 723

### 24.3.28 DMA burst mode
- Page 723

### 24.3.29 Debug mode
- Page 724

### 24.4 TIM1 registers
- Page 725

#### 24.4.1 TIM1 control register 1 (TIM1_CR1)
- Page 725

#### 24.4.2 TIM1 control register 2 (TIM1_CR2)
- Page 726

#### 24.4.3 TIM1 slave mode control register (TIM1_SMCR)
- Page 729

#### 24.4.4 TIM1 DMA/interrupt enable register (TIM1_DIER)
- Page 731

#### 24.4.5 TIM1 status register (TIM1_SR)
- Page 733

#### 24.4.6 TIM1 event generation register (TIM1_EGR)
- Page 735

#### 24.4.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1)
- Page 736

#### 24.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)
- Page 737

#### 24.4.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2)
- Page 740

#### 24.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)
- Page 741

#### 24.4.11 TIM1 capture/compare enable register (TIM1_CCER)
- Page 743

#### 24.4.12 TIM1 counter (TIM1_CNT)
- Page 746

#### 24.4.13 TIM1 prescaler (TIM1_PSC)
- Page 746

#### 24.4.14 TIM1 auto-reload register (TIM1_ARR)
- Page 746

#### 24.4.15 TIM1 repetition counter register (TIM1_RCR)
- Page 747

#### 24.4.16 TIM1 capture/compare register 1 (TIM1_CCR1)
- Page 747
24.4.17 TIM1 capture/compare register 2 
(TIM1_CCR2) .................................................. 748
24.4.18 TIM1 capture/compare register 3 
(TIM1_CCR3) .................................................. 748
24.4.19 TIM1 capture/compare register 4 
(TIM1_CCR4) .................................................. 749
24.4.20 TIM1 break and dead-time register 
(TIM1_BDTR) .................................................. 749
24.4.21 TIM1 DMA control register 
(TIM1_DCR) .................................................. 753
24.4.22 TIM1 DMA address for full transfer 
(TIM1_DMAR) .................................................. 754
24.4.23 TIM1 option register 1 (TIM1_OR1) .................................................. 755
24.4.24 TIM1 capture/compare mode register 3 
(TIM1_CCMR3) .................................................. 755
24.4.25 TIM1 capture/compare register 5 
(TIM1_CCR5) .................................................. 756
24.4.26 TIM1 capture/compare register 6 
(TIM1_CCR6) .................................................. 757
24.4.27 TIM1 alternate function option register 1 (TIM1_AF1) .................................................. 758
24.4.28 TIM1 Alternate function register 2 (TIM1_AF2) .................................................. 759
24.4.29 TIM1 timer input selection register (TIM1_TISEL) .................................................. 761
24.4.30 TIM1 register map .................................................. 762

25 General-purpose timer (TIM2) .................................................. 765
25.1 TIM2 introduction .................................................. 765
25.2 TIM2 main features .................................................. 765
25.3 TIM2 functional description .................................................. 767
  25.3.1 Time-base unit .................................................. 767
  25.3.2 Counter modes .................................................. 769
  25.3.3 Clock selection .................................................. 779
  25.3.4 Capture/Compare channels .................................................. 783
  25.3.5 Input capture mode .................................................. 785
  25.3.6 PWM input mode .................................................. 786
  25.3.7 Forced output mode .................................................. 787
  25.3.8 Output compare mode .................................................. 787
  25.3.9 PWM mode .................................................. 788
  25.3.10 Asymmetric PWM mode .................................................. 792
  25.3.11 Combined PWM mode .................................................. 792
25.3.12 Clearing the OCxREF signal on an external event .......................... 793
25.3.13 One-pulse mode .................................................. 795
25.3.14 Retriggerable one pulse mode ..................................... 796
25.3.15 Encoder interface mode .......................................... 797
25.3.16 UIF bit remapping .................................................. 799
25.3.17 Timer input XOR function ....................................... 799
25.3.18 Timers and external trigger synchronization ...................... 800
25.3.19 Timer synchronization ............................................. 803
25.3.20 DMA burst mode .................................................. 808
25.3.21 Debug mode ...................................................... 809

25.4 TIM2 registers ....................................................... 810
25.4.1 TIM2 control register 1 (TIM2_CR1) ............................... 810
25.4.2 TIM2 control register 2 (TIM2_CR2) ............................... 811
25.4.3 TIM2 slave mode control register (TIM2_SMCR) .................... 813
25.4.4 TIM2 DMA/Interrupt enable register (TIM2_DIER) ................. 816
25.4.5 TIM2 status register (TIM2_SR) .................................. 817
25.4.6 TIM2 event generation register (TIM2_EGR) ...................... 819
25.4.7 TIM2 capture/compare mode register 1 (TIM2_CCMR1) .......... 820
25.4.8 TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1) 822
25.4.9 TIM2 capture/compare mode register 2 (TIM2_CCMR2) .......... 824
25.4.10 TIM2 capture/compare mode register 2 [alternate] (TIM2_CCMR2) 825
25.4.11 TIM2 capture/compare enable register (TIM2_CCER) ............. 826
25.4.12 TIM2 counter (TIM2_CNT) ...................................... 827
25.4.13 TIM2 counter [alternate] (TIM2_CNT) ............................ 828
25.4.14 TIM2 prescaler (TIM2_PSC) ..................................... 828
25.4.15 TIM2 auto-reload register (TIM2_ARR) .......................... 829
25.4.16 TIM2 capture/compare register 1 (TIM2_CCR1) .................. 829
25.4.17 TIM2 capture/compare register 2 (TIM2_CCR2) .................. 829
25.4.18 TIM2 capture/compare register 3 (TIM2_CCR3) ................. 830
25.4.19 TIM2 capture/compare register 4 (TIM2_CCR4) ................. 830
25.4.20 TIM2 DMA control register (TIM2_DCR) .......................... 831
25.4.21 TIM2 DMA address for full transfer (TIM2_DMAR) ............... 832
25.4.22 TIM2 option register 1 (TIM2_OR1) .............................. 832
25.4.23 TIM2 alternate function option register 1 (TIM2_AF1) ......... 832
25.4.24 TIM2 timer input selection register (TIM2_TISEL) ............. 833
25.4.25 TIMx register map .............................................. 834
26 General-purpose timers (TIM16/TIM17) ........................................... 837
  26.1 TIM16/TIM17 introduction ................................................. 837
  26.2 TIM16/TIM17 main features .............................................. 837
  26.3 TIM16/TIM17 functional description ..................................... 839
    26.3.1 Time-base unit ..................................................... 839
    26.3.2 Counter modes ..................................................... 841
    26.3.3 Repetition counter ................................................ 845
    26.3.4 Clock selection ................................................... 846
    26.3.5 Capture/compare channels ....................................... 848
    26.3.6 Input capture mode .............................................. 850
    26.3.7 Forced output mode .............................................. 851
    26.3.8 Output compare mode ............................................ 851
    26.3.9 PWM mode ......................................................... 853
    26.3.10 Complementary outputs and dead-time insertion .............. 854
    26.3.11 Using the break function ..................................... 856
    26.3.12 Bidirectional break inputs ................................... 859
    26.3.13 6-step PWM generation ......................................... 860
    26.3.14 One-pulse mode ................................................. 862
    26.3.15 UIF bit remapping .............................................. 863
    26.3.16 Slave mode – combined reset + trigger mode .................. 863
    26.3.17 DMA burst mode ................................................. 863
    26.3.18 Using timer output as trigger for other timers (TIM16/TIM17) . . 864
    26.3.19 Debug mode ..................................................... 865
  26.4 TIM16/TIM17 registers .................................................... 866
    26.4.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) ............. 866
    26.4.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) ............. 867
    26.4.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) .... 868
    26.4.4 TIMx status register (TIMx_SR)(x = 16 to 17) .................. 869
    26.4.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) ....... 870
    26.4.6 TIMx capture/compare mode register 1
      (TIMx_CCMR1)(x = 16 to 17) ....................................... 871
    26.4.7 TIMx capture/compare mode register 1 [alternate]
      (TIMx_CCMR1)(x = 16 to 17) ....................................... 872
    26.4.8 TIMx capture/compare enable register (TIMx_CCRER)(x = 16 to 17) . 874
    26.4.9 TIMx counter (TIMx_CNT)(x = 16 to 17) .......................... 876
    26.4.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) ...................... 877
    26.4.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) .......... 877
26.4.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) ........ 878
26.4.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) ...... 878
26.4.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) .... 879
26.4.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) ............. 881
26.4.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) ... 882
26.4.17 TIM16 option register 1 (TIM16_OR1) .................. 883
26.4.18 TIM16 alternate function register 1 (TIM16_AF1) ...................... 883
26.4.19 TIM16 input selection register (TIM16_TISEL) ...................... 884
26.4.20 TIM17 option register 1 (TIM17_OR1) .................. 885
26.4.21 TIM17 alternate function register 1 (TIM17_AF1) ...................... 885
26.4.22 TIM17 input selection register (TIM17_TISEL) ...................... 886
26.4.23 TIM16/TIM17 register map .......................... 887

27 Low-power timer (LPTIM) ........................................... 889

27.1 Introduction ............................................ 889
27.2 LPTIM main features ........................................... 889
27.3 LPTIM implementation ........................................ 890
27.4 LPTIM functional description ................................. 891
  27.4.1 LPTIM block diagram ................................... 891
  27.4.2 LPTIM trigger mapping .................................. 891
  27.4.3 LPTIM reset and clocks .................................. 892
  27.4.4 Glitch filter .......................................... 892
  27.4.5 Prescaler ........................................... 893
  27.4.6 Trigger multiplexer ..................................... 894
  27.4.7 Operating mode ........................................ 894
  27.4.8 Timeout function ....................................... 896
  27.4.9 Waveform generation .................................... 896
  27.4.10 Register update ....................................... 897
  27.4.11 Counter mode ........................................ 898
  27.4.12 Timer enable .......................................... 898
  27.4.13 Timer counter reset .................................... 899
  27.4.14 Encoder mode ........................................ 899
  27.4.15 Debug mode ........................................ 901

27.5 LPTIM low-power modes ....................................... 901
27.6 LPTIM interrupts ........................................... 902
27.7 LPTIM registers ............................................ 902
27.7.1 LPTIM interrupt and status register (LPTIM_ISR) ........................................... 903
27.7.2 LPTIM interrupt clear register (LPTIM_ICR) ....................................................... 904
27.7.3 LPTIM interrupt enable register (LPTIM_IER) ..................................................... 904
27.7.4 LPTIM configuration register (LPTIM_CFGR) ....................................................... 905
27.7.5 LPTIM control register (LPTIM_CR) ................................................................. 908
27.7.6 LPTIM compare register (LPTIM_CMP) ............................................................. 910
27.7.7 LPTIM autoreload register (LPTIM_ARR) ............................................................ 910
27.7.8 LPTIM counter register (LPTIM_CNT) ............................................................... 911
27.7.9 LPTIM1 option register (LPTIM1_OR) ............................................................... 911
27.7.10 LPTIM2 option register (LPTIM2_OR) .............................................................. 912
27.7.11 LPTIM register map ................................ ............................................................. 913

28 Infrared interface (IRTIM) ........................................... 914

29 Real-time clock (RTC) ........................................... 915
29.1 Introduction ................................................................. 915
29.2 RTC main features ........................................................... 916
29.3 RTC implementation ........................................................... 916
29.4 RTC functional description .................................................. 917
  29.4.1 RTC block diagram ................................................................. 917
  29.4.2 Clock and prescalers ................................................................. 918
  29.4.3 Real-time clock and calendar ....................................................... 918
  29.4.4 Programmable alarms ................................................................. 919
  29.4.5 Periodic auto-wake-up ................................................................. 919
  29.4.6 RTC initialization and configuration .................................................. 920
  29.4.7 Reading the calendar ................................................................. 921
  29.4.8 Resetting the RTC ................................................................. 922
  29.4.9 RTC synchronization ................................................................. 923
  29.4.10 RTC reference clock detection .................................................. 923
  29.4.11 RTC smooth digital calibration .................................................. 924
  29.4.12 Time-stamp function ................................................................. 926
  29.4.13 Tamper detection ................................................................. 927
  29.4.14 Calibration clock output ........................................................... 929
  29.4.15 Alarm output ................................................................. 929
29.5 RTC low-power modes .................................................. 930
29.6 RTC interrupts ............................................................... 930
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>29.7</td>
<td>RTC registers</td>
<td>931</td>
</tr>
<tr>
<td>29.7.1</td>
<td>RTC time register (RTC_TR)</td>
<td>931</td>
</tr>
<tr>
<td>29.7.2</td>
<td>RTC date register (RTC_DR)</td>
<td>932</td>
</tr>
<tr>
<td>29.7.3</td>
<td>RTC control register (RTC_CR)</td>
<td>933</td>
</tr>
<tr>
<td>29.7.4</td>
<td>RTC initialization and status register (RTC_ISR)</td>
<td>936</td>
</tr>
<tr>
<td>29.7.5</td>
<td>RTC prescaler register (RTC_PRER)</td>
<td>939</td>
</tr>
<tr>
<td>29.7.6</td>
<td>RTC wake-up timer register (RTC_WUTR)</td>
<td>940</td>
</tr>
<tr>
<td>29.7.7</td>
<td>RTC alarm A register (RTC_ALRMAR)</td>
<td>941</td>
</tr>
<tr>
<td>29.7.8</td>
<td>RTC alarm B register (RTC_ALRMBR)</td>
<td>942</td>
</tr>
<tr>
<td>29.7.9</td>
<td>RTC write protection register (RTC_WPR)</td>
<td>943</td>
</tr>
<tr>
<td>29.7.10</td>
<td>RTC sub second register (RTC_SSR)</td>
<td>943</td>
</tr>
<tr>
<td>29.7.11</td>
<td>RTC shift control register (RTC_SHIFTR)</td>
<td>944</td>
</tr>
<tr>
<td>29.7.12</td>
<td>RTC timestamp time register (RTC_TSTR)</td>
<td>945</td>
</tr>
<tr>
<td>29.7.13</td>
<td>RTC timestamp date register (RTC_TSDR)</td>
<td>946</td>
</tr>
<tr>
<td>29.7.14</td>
<td>RTC time-stamp sub second register (RTC_TSSSR)</td>
<td>947</td>
</tr>
<tr>
<td>29.7.15</td>
<td>RTC calibration register (RTC_CALR)</td>
<td>948</td>
</tr>
<tr>
<td>29.7.16</td>
<td>RTC tamper configuration register (RTC_TAMPCR)</td>
<td>949</td>
</tr>
<tr>
<td>29.7.17</td>
<td>RTC alarm A sub second register (RTC_ALRMASSR)</td>
<td>952</td>
</tr>
<tr>
<td>29.7.18</td>
<td>RTC alarm B sub second register (RTC_ALRMBSSR)</td>
<td>953</td>
</tr>
<tr>
<td>29.7.19</td>
<td>RTC option register (RTC_OR)</td>
<td>954</td>
</tr>
<tr>
<td>29.7.20</td>
<td>RTC backup registers (RTC_BKPxR)</td>
<td>954</td>
</tr>
<tr>
<td>29.7.21</td>
<td>RTC register map</td>
<td>955</td>
</tr>
<tr>
<td>30</td>
<td>Independent watchdog (IWDG)</td>
<td>957</td>
</tr>
<tr>
<td>30.1</td>
<td>Introduction</td>
<td>957</td>
</tr>
<tr>
<td>30.2</td>
<td>IWDG main features</td>
<td>957</td>
</tr>
<tr>
<td>30.3</td>
<td>IWDG functional description</td>
<td>957</td>
</tr>
<tr>
<td>30.3.1</td>
<td>IWDG block diagram</td>
<td>957</td>
</tr>
<tr>
<td>30.3.2</td>
<td>Window option</td>
<td>958</td>
</tr>
<tr>
<td>30.3.3</td>
<td>Hardware watchdog</td>
<td>959</td>
</tr>
<tr>
<td>30.3.4</td>
<td>Low-power freeze</td>
<td>959</td>
</tr>
<tr>
<td>30.3.5</td>
<td>Register access protection</td>
<td>959</td>
</tr>
<tr>
<td>30.3.6</td>
<td>Debug mode</td>
<td>959</td>
</tr>
<tr>
<td>30.4</td>
<td>IWDG registers</td>
<td>960</td>
</tr>
<tr>
<td>30.4.1</td>
<td>IWDG key register (IWDG_KR)</td>
<td>960</td>
</tr>
<tr>
<td>30.4.2</td>
<td>IWDG prescaler register (IWDG_PR)</td>
<td>961</td>
</tr>
<tr>
<td>30.4.3</td>
<td>IWDG reload register (IWDG_RLR)</td>
<td>962</td>
</tr>
</tbody>
</table>
31 System window watchdog (WWDG) .................................. 966
  31.1 Introduction .................................................... 966
  31.2 WWDG main features .......................................... 966
  31.3 WWDG functional description ................................. 966
    31.3.1 WWDG block diagram ................................ 967
    31.3.2 Enabling the watchdog ................................ 967
    31.3.3 Controlling the down-counter ........................ 967
    31.3.4 How to program the watchdog timeout ............... 967
    31.3.5 Debug mode ............................................. 969
  31.4 WWDG interrupts ............................................. 969
  31.5 WWDG registers .............................................. 969
    31.5.1 WWDG control register (WWDG_CR) .................. 969
    31.5.2 WWDG configuration register (WWDG_CFR) ........... 970
    31.5.3 WWDG status register (WWDG_SR) ................... 971
    31.5.4 WWDG register map ................................... 971

32 Inter-integrated circuit (I2C) interface ...................... 972
  32.1 Introduction ................................................ 972
  32.2 I2C main features ........................................... 972
  32.3 I2C implementation .......................................... 973
  32.4 I2C functional description ................................ 973
    32.4.1 I2C block diagram .................................. 974
    32.4.2 I2C pins and internal signals ...................... 974
    32.4.3 I2C clock requirements ............................ 975
    32.4.4 Mode selection ..................................... 975
    32.4.5 I2C initialization .................................. 976
    32.4.6 I2C reset .......................................... 980
    32.4.7 Data transfer ....................................... 981
    32.4.8 I2C slave mode ................................... 983
    32.4.9 I2C master mode .................................. 992
    32.4.10 I2C_TIMINGR register configuration examples .... 1003
    32.4.11 SMBus specific features ............................ 1005
32.4.12 SMBus initialization ........................................ 1007
32.4.13 SMBus I2C_TIMEOUTR register configuration examples .... 1009
32.4.14 SMBus slave mode ........................................ 1009
32.4.15 SMBus master mode ...................................... 1013
32.4.16 Wake-up from Stop mode on address match .................. 1016
32.4.17 Error conditions ........................................ 1017
32.5 I2C in low-power modes ..................................... 1019
32.6 I2C interrupts ............................................... 1019
32.7 I2C DMA requests ........................................... 1020
32.8 I2C debug modes ........................................... 1020
32.9 I2C registers ............................................... 1021
  32.9.1 I2C control register 1 (I2C_CR1) ......................... 1021
  32.9.2 I2C control register 2 (I2C_CR2) ......................... 1023
  32.9.3 I2C own address 1 register (I2C_OAR1) .................. 1025
  32.9.4 I2C own address 2 register (I2C_OAR2) .................. 1026
  32.9.5 I2C timing register (I2C_TIMINGR) ...................... 1027
  32.9.6 I2C timeout register (I2C_TIMEOUTR) ................... 1028
  32.9.7 I2C interrupt and status register (I2C_ISR) ............... 1029
  32.9.8 I2C interrupt clear register (I2C_ICR) .................. 1031
  32.9.9 I2C PEC register (I2C_PECR) .......................... 1032
  32.9.10 I2C receive data register (I2C_RXDR) ................... 1032
  32.9.11 I2C transmit data register (I2C_TXDR) ................... 1033
  32.9.12 I2C register map ...................................... 1034

33 Universal synchronous/asynchronous receiver
transmitter (USART/UART) ....................................... 1036
  33.1 USART introduction ...................................... 1036
  33.2 USART main features ..................................... 1037
  33.3 USART extended features ................................. 1038
  33.4 USART implementation ................................. 1038
  33.5 USART functional description ............................ 1039
    33.5.1 USART block diagram ............................... 1039
    33.5.2 USART signals ..................................... 1040
    33.5.3 USART character description ......................... 1041
    33.5.4 USART FIFOs and thresholds ......................... 1043
    33.5.5 USART transmitter .................................. 1043
## 33.5 USART functions

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.5.6</td>
<td>USART receiver</td>
<td>1047</td>
</tr>
<tr>
<td>33.5.7</td>
<td>USART baud rate generation</td>
<td>1054</td>
</tr>
<tr>
<td>33.5.8</td>
<td>Tolerance of the USART receiver to clock deviation</td>
<td>1055</td>
</tr>
<tr>
<td>33.5.9</td>
<td>USART auto baud rate detection</td>
<td>1057</td>
</tr>
<tr>
<td>33.5.10</td>
<td>USART multiprocessor communication</td>
<td>1059</td>
</tr>
<tr>
<td>33.5.11</td>
<td>USART Modbus communication</td>
<td>1061</td>
</tr>
<tr>
<td>33.5.12</td>
<td>USART parity control</td>
<td>1062</td>
</tr>
<tr>
<td>33.5.13</td>
<td>USART LIN (local interconnection network) mode</td>
<td>1063</td>
</tr>
<tr>
<td>33.5.14</td>
<td>USART synchronous mode</td>
<td>1065</td>
</tr>
<tr>
<td>33.5.15</td>
<td>USART single-wire half-duplex communication</td>
<td>1069</td>
</tr>
<tr>
<td>33.5.16</td>
<td>USART receiver timeout</td>
<td>1069</td>
</tr>
<tr>
<td>33.5.17</td>
<td>USART smartcard mode</td>
<td>1070</td>
</tr>
<tr>
<td>33.5.18</td>
<td>USART IrDA SIR ENDEC block</td>
<td>1074</td>
</tr>
<tr>
<td>33.5.19</td>
<td>Continuous communication using USART and DMA</td>
<td>1077</td>
</tr>
<tr>
<td>33.5.20</td>
<td>RS232 hardware flow control and RS485 Driver Enable</td>
<td>1079</td>
</tr>
<tr>
<td>33.5.21</td>
<td>USART low-power management</td>
<td>1082</td>
</tr>
</tbody>
</table>

### 33.6 USART in low-power modes

- USART in low-power modes | 1085

### 33.7 USART interrupts

- USART interrupts | 1086

### 33.8 USART registers

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>33.8.1</td>
<td>USART control register 1 (USART_CR1)</td>
<td>1087</td>
</tr>
<tr>
<td>33.8.2</td>
<td>USART control register 1 [alternate] (USART_CR1)</td>
<td>1091</td>
</tr>
<tr>
<td>33.8.3</td>
<td>USART control register 2 (USART_CR2)</td>
<td>1094</td>
</tr>
<tr>
<td>33.8.4</td>
<td>USART control register 3 (USART_CR3)</td>
<td>1098</td>
</tr>
<tr>
<td>33.8.5</td>
<td>USART baud rate register (USART_BRR)</td>
<td>1103</td>
</tr>
<tr>
<td>33.8.6</td>
<td>USART guard time and prescaler register (USART_GTPR)</td>
<td>1103</td>
</tr>
<tr>
<td>33.8.7</td>
<td>USART receiver timeout register (USART_RTOR)</td>
<td>1104</td>
</tr>
<tr>
<td>33.8.8</td>
<td>USART request register (USART_RQR)</td>
<td>1105</td>
</tr>
<tr>
<td>33.8.9</td>
<td>USART interrupt and status register (USART_ISR)</td>
<td>1106</td>
</tr>
<tr>
<td>33.8.10</td>
<td>USART interrupt and status register [alternate] (USART_ISR)</td>
<td>1112</td>
</tr>
<tr>
<td>33.8.11</td>
<td>USART interrupt flag clear register (USART_ICR)</td>
<td>1117</td>
</tr>
<tr>
<td>33.8.12</td>
<td>USART receive data register (USART_RDR)</td>
<td>1119</td>
</tr>
<tr>
<td>33.8.13</td>
<td>USART transmit data register (USART_TDR)</td>
<td>1119</td>
</tr>
<tr>
<td>33.8.14</td>
<td>USART prescaler register (USART_PRESC)</td>
<td>1120</td>
</tr>
<tr>
<td>33.8.15</td>
<td>USART register map</td>
<td>1121</td>
</tr>
</tbody>
</table>

### 34 Low-power universal asynchronous receiver transmitter (LPUART)

- Low-power universal asynchronous receiver transmitter (LPUART) | 1123
34.1 LPUART introduction ............................................. 1123
34.2 LPUART main features ........................................... 1124
34.3 LPUART implementation ........................................... 1125
34.4 LPUART functional description ................................. 1126
  34.4.1 LPUART block diagram ................................. 1126
  34.4.2 LPUART signals ........................................... 1127
  34.4.3 LPUART character description ......................... 1128
  34.4.4 LPUART FIFOs and thresholds ......................... 1129
  34.4.5 LPUART transmitter ..................................... 1130
  34.4.6 LPUART receiver ........................................ 1133
  34.4.7 LPUART baud rate generation ......................... 1137
  34.4.8 Tolerance of the LPUART receiver to clock deviation 1138
  34.4.9 LPUART multiprocessor communication ............... 1139
  34.4.10 LPUART parity control ............................... 1141
  34.4.11 LPUART single-wire half-duplex communication .... 1142
  34.4.12 Continuous communication using DMA and LPUART . 1142
  34.4.13 RS232 hardware flow control and RS485 Driver Enable 1145
  34.4.14 LPUART low-power management ...................... 1147
34.5 LPUART in low-power modes .................................. 1150
34.6 LPUART interrupts ............................................ 1151
34.7 LPUART registers ............................................... 1152
  34.7.1 LPUART control register 1 (LPUART_CR1) ............... 1152
  34.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 1155
  34.7.3 LPUART control register 2 (LPUART_CR2) ............... 1158
  34.7.4 LPUART control register 3 (LPUART_CR3) ............... 1160
  34.7.5 LPUART baud rate register (LPUART_BRR) ............. 1163
  34.7.6 LPUART request register (LPUART_RQR) ............... 1164
  34.7.7 LPUART interrupt and status register (LPUART_ISR) .... 1164
  34.7.8 LPUART interrupt and status register [alternate] (LPUART_ISR) 1169
  34.7.9 LPUART interrupt flag clear register (LPUART_ICR) .... 1172
  34.7.10 LPUART receive data register (LPUART_RDR) ......... 1173
  34.7.11 LPUART transmit data register (LPUART_TDR) ........ 1173
  34.7.12 LPUART prescaler register (LPUART_PRESC) ........ 1174
  34.7.13 LPUART register map .................................. 1175

35 Serial peripheral interface (SPI) ................................ 1177
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.1</td>
<td>Introduction</td>
<td>1177</td>
</tr>
<tr>
<td>35.2</td>
<td>SPI main features</td>
<td>1177</td>
</tr>
<tr>
<td>35.3</td>
<td>SPI implementation</td>
<td>1178</td>
</tr>
<tr>
<td>35.4</td>
<td>SPI functional description</td>
<td>1178</td>
</tr>
<tr>
<td>35.4.1</td>
<td>General description</td>
<td>1178</td>
</tr>
<tr>
<td>35.4.2</td>
<td>Communications between one master and one slave</td>
<td>1179</td>
</tr>
<tr>
<td>35.4.3</td>
<td>Standard multislave communication</td>
<td>1181</td>
</tr>
<tr>
<td>35.4.4</td>
<td>Multimaster communication</td>
<td>1182</td>
</tr>
<tr>
<td>35.4.5</td>
<td>Slave select (NSS) pin management</td>
<td>1183</td>
</tr>
<tr>
<td>35.4.6</td>
<td>Communication formats</td>
<td>1184</td>
</tr>
<tr>
<td>35.4.7</td>
<td>Configuration of SPI</td>
<td>1186</td>
</tr>
<tr>
<td>35.4.8</td>
<td>Procedure for enabling SPI</td>
<td>1187</td>
</tr>
<tr>
<td>35.4.9</td>
<td>Data transmission and reception procedures</td>
<td>1187</td>
</tr>
<tr>
<td>35.4.10</td>
<td>SPI status flags</td>
<td>1197</td>
</tr>
<tr>
<td>35.4.11</td>
<td>SPI error flags</td>
<td>1198</td>
</tr>
<tr>
<td>35.4.12</td>
<td>NSS pulse mode</td>
<td>1199</td>
</tr>
<tr>
<td>35.4.13</td>
<td>TI mode</td>
<td>1199</td>
</tr>
<tr>
<td>35.4.14</td>
<td>CRC calculation</td>
<td>1200</td>
</tr>
<tr>
<td>35.5</td>
<td>SPI interrupts</td>
<td>1202</td>
</tr>
<tr>
<td>35.6</td>
<td>SPI registers</td>
<td>1203</td>
</tr>
<tr>
<td>35.6.1</td>
<td>SPI control register 1 (SPIx_CR1)</td>
<td>1203</td>
</tr>
<tr>
<td>35.6.2</td>
<td>SPI control register 2 (SPIx_CR2)</td>
<td>1205</td>
</tr>
<tr>
<td>35.6.3</td>
<td>SPI status register (SPIx_SR)</td>
<td>1207</td>
</tr>
<tr>
<td>35.6.4</td>
<td>SPI data register (SPIx_DR)</td>
<td>1208</td>
</tr>
<tr>
<td>35.6.5</td>
<td>SPI CRC polynomial register (SPIx_CRCPR)</td>
<td>1209</td>
</tr>
<tr>
<td>35.6.6</td>
<td>SPI Rx CRC register (SPIx_RXCRCR)</td>
<td>1209</td>
</tr>
<tr>
<td>35.6.7</td>
<td>SPI Tx CRC register (SPIx_TXCRCR)</td>
<td>1209</td>
</tr>
<tr>
<td>35.6.8</td>
<td>SPI register map</td>
<td>1211</td>
</tr>
<tr>
<td>36</td>
<td>Serial audio interface (SAI)</td>
<td>1212</td>
</tr>
<tr>
<td>36.1</td>
<td>Introduction</td>
<td>1212</td>
</tr>
<tr>
<td>36.2</td>
<td>SAI main features</td>
<td>1212</td>
</tr>
<tr>
<td>36.3</td>
<td>SAI implementation</td>
<td>1213</td>
</tr>
<tr>
<td>36.4</td>
<td>SAI functional description</td>
<td>1213</td>
</tr>
<tr>
<td>36.4.1</td>
<td>SAI block diagram</td>
<td>1213</td>
</tr>
<tr>
<td>36.4.2</td>
<td>SAI pins and internal signals</td>
<td>1215</td>
</tr>
</tbody>
</table>
36.4.3 Main SAI modes ...................................................... 1215
36.4.4 SAI synchronization mode ........................................ 1216
36.4.5 Audio data size .................................................... 1217
36.4.6 Frame synchronization ............................................ 1217
36.4.7 Slot configuration .................................................. 1220
36.4.8 SAI clock generator ............................................... 1222
36.4.9 Internal FIFOs ..................................................... 1225
36.4.10 PDM interface ..................................................... 1227
36.4.11 AC’97 link controller ............................................. 1235
36.4.12 SPDIF output ...................................................... 1236
36.4.13 Specific features .................................................. 1239
36.4.14 Error flags ......................................................... 1243
36.4.15 Disabling the SAI ................................................ 1246
36.4.16 SAI DMA interface ............................................... 1246
36.5 SAI interrupts ........................................................ 1247
36.6 SAI registers ........................................................ 1248
  36.6.1 SAI configuration register 1 (SAI_ACR1) ....................... 1248
  36.6.2 SAI configuration register 2 (SAI_ACR2) ....................... 1250
  36.6.3 SAI frame configuration register (SAI_AFRCR) ............... 1252
  36.6.4 SAI slot register (SAI_ASLOTR) ............................... 1253
  36.6.5 SAI interrupt mask register (SAI_AIM) ....................... 1254
  36.6.6 SAI status register (SAI_ASR) .................................. 1256
  36.6.7 SAI clear flag register (SAI_ACLRFR) ......................... 1258
  36.6.8 SAI data register (SAI_ADR) ................................... 1259
  36.6.9 SAI configuration register 1 (SAI_BCR1) ....................... 1259
  36.6.10 SAI configuration register 2 (SAI_BCR2) ...................... 1262
  36.6.11 SAI frame configuration register (SAI_BFRCR) .............. 1264
  36.6.12 SAI slot register (SAI_BSLOTR) .............................. 1265
  36.6.13 SAI interrupt mask register (SAI_BIM) ....................... 1266
  36.6.14 SAI status register (SAI_BSR) ................................ 1267
  36.6.15 SAI clear flag register (SAI_BCLRFR) ....................... 1269
  36.6.16 SAI data register (SAI_BDR) .................................. 1270
  36.6.17 SAI PDM control register (SAI_PDMCR) ...................... 1271
  36.6.18 SAI PDM delay register (SAI_PDMDLY) ....................... 1272
  36.6.19 SAI register map ............................................... 1274

37 Inter-processor communication controller (IPCC) .................... 1276
37.1 Introduction ......................................................... 1276
37.2 IPCC main features ............................................. 1276
37.3 IPCC functional description ................................. 1276
  37.3.1 IPCC block diagram ....................................... 1277
  37.3.2 IPCC Simplex channel mode ............................... 1277
  37.3.3 IPCC Half-duplex channel mode ......................... 1280
  37.3.4 IPCC interrupts ........................................... 1283
37.4 IPCC registers .................................................. 1284
  37.4.1 IPCC processor 1 control register (IPCC_C1CR) .... 1284
  37.4.2 IPCC processor 1 mask register (IPCC_C1MR) ........ 1284
  37.4.3 IPCC processor 1 status set clear register (IPCC_C1SCR) 1285
  37.4.4 IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR) 1286
  37.4.5 IPCC processor 2 control register (IPCC_C2CR) .... 1286
  37.4.6 IPCC processor 2 mask register (IPCC_C2MR) ........ 1287
  37.4.7 IPCC processor 2 status set clear register (IPCC_C2SCR) 1287
  37.4.8 IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR) 1288
  37.4.9 IPCC register map ........................................ 1289

38 Hardware semaphore (HSEM) ................................. 1290
38.1 Introduction .................................................. 1290
38.2 Main features ................................................. 1290
38.3 Functional description ....................................... 1291
  38.3.1 HSEM block diagram ..................................... 1291
  38.3.2 HSEM internal signals ................................... 1291
  38.3.3 HSEM lock procedures ................................... 1291
  38.3.4 HSEM write/read/read lock register address .......... 1293
  38.3.5 HSEM unlock procedures .................................. 1293
  38.3.6 HSEM COREID semaphore clear ......................... 1294
  38.3.7 HSEM interrupts ......................................... 1294
  38.3.8 AHB bus master ID verification ......................... 1296
38.4 HSEM registers ................................................. 1297
  38.4.1 HSEM register semaphore x (HSEM_Rx) ................. 1297
  38.4.2 HSEM read lock register semaphore x (HSEM_RLRx) .... 1298
  38.4.3 HSEM interrupt enable register (HSEM_CnIER) ....... 1299
  38.4.4 HSEM interrupt clear register (HSEM_CnICR) ......... 1299
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>38.4.5</td>
<td>HSEM interrupt status register (HSEM_CnISR)</td>
<td>1299</td>
</tr>
<tr>
<td>38.4.6</td>
<td>HSEM interrupt status register (HSEM_CnMISR)</td>
<td>1300</td>
</tr>
<tr>
<td>38.4.7</td>
<td>HSEM clear register (HSEM_CR)</td>
<td>1300</td>
</tr>
<tr>
<td>38.4.8</td>
<td>HSEM clear semaphore key register (HSEM_KEYR)</td>
<td>1301</td>
</tr>
<tr>
<td>38.4.9</td>
<td>HSEM register map</td>
<td>1302</td>
</tr>
<tr>
<td>39</td>
<td>Universal serial bus full-speed device interface (USB)</td>
<td>1304</td>
</tr>
<tr>
<td>39.1</td>
<td>Introduction</td>
<td>1304</td>
</tr>
<tr>
<td>39.2</td>
<td>USB main features</td>
<td>1304</td>
</tr>
<tr>
<td>39.3</td>
<td>USB implementation</td>
<td>1304</td>
</tr>
<tr>
<td>39.4</td>
<td>USB functional description</td>
<td>1305</td>
</tr>
<tr>
<td>39.4.1</td>
<td>Description of USB blocks</td>
<td>1306</td>
</tr>
<tr>
<td>39.5</td>
<td>Programming considerations</td>
<td>1307</td>
</tr>
<tr>
<td>39.5.1</td>
<td>Generic USB device programming</td>
<td>1307</td>
</tr>
<tr>
<td>39.5.2</td>
<td>System and power-on reset</td>
<td>1308</td>
</tr>
<tr>
<td>39.5.3</td>
<td>Double-buffered endpoints</td>
<td>1313</td>
</tr>
<tr>
<td>39.5.4</td>
<td>Isochronous transfers</td>
<td>1315</td>
</tr>
<tr>
<td>39.5.5</td>
<td>Suspend/Resume events</td>
<td>1316</td>
</tr>
<tr>
<td>39.6</td>
<td>USB and USB SRAM registers</td>
<td>1319</td>
</tr>
<tr>
<td>39.6.1</td>
<td>Common registers</td>
<td>1319</td>
</tr>
<tr>
<td>39.6.2</td>
<td>Buffer descriptor table</td>
<td>1332</td>
</tr>
<tr>
<td>39.6.3</td>
<td>USB register map</td>
<td>1335</td>
</tr>
<tr>
<td>40</td>
<td>Clock recovery system (CRS)</td>
<td>1337</td>
</tr>
<tr>
<td>40.1</td>
<td>CRS introduction</td>
<td>1337</td>
</tr>
<tr>
<td>40.2</td>
<td>CRS main features</td>
<td>1337</td>
</tr>
<tr>
<td>40.3</td>
<td>CRS implementation</td>
<td>1337</td>
</tr>
<tr>
<td>40.4</td>
<td>CRS functional description</td>
<td>1338</td>
</tr>
<tr>
<td>40.4.1</td>
<td>CRS block diagram</td>
<td>1338</td>
</tr>
<tr>
<td>40.4.2</td>
<td>Synchronization input</td>
<td>1338</td>
</tr>
<tr>
<td>40.4.3</td>
<td>Frequency error measurement</td>
<td>1338</td>
</tr>
<tr>
<td>40.4.4</td>
<td>Frequency error evaluation and automatic trimming</td>
<td>1339</td>
</tr>
<tr>
<td>40.4.5</td>
<td>CRS initialization and configuration</td>
<td>1340</td>
</tr>
<tr>
<td>40.5</td>
<td>CRS low-power modes</td>
<td>1341</td>
</tr>
<tr>
<td>40.6</td>
<td>CRS interrupts</td>
<td>1341</td>
</tr>
<tr>
<td>40.7</td>
<td>CRS registers</td>
<td>1342</td>
</tr>
</tbody>
</table>
### Debug support (DBG) .................................................. 1348

#### 41.1 Introduction .................................................. 1348

#### 41.2 Debug use cases ............................................... 1348

#### 41.3 DBG functional description ................................. 1350

- 41.3.1 DBG block diagram .......................................... 1350
- 41.3.2 DBG pins and internal signals ............................... 1350
- 41.3.3 DBG power domains .......................................... 1351
- 41.3.4 DBG clocks .................................................. 1351
- 41.3.5 Debug and low power modes ................................ 1352
- 41.3.6 DBG reset ................................................... 1352

#### 41.4 Serial wire and JTAG debug port (SWJ-DP) ................. 1352

- 41.4.1 JTAG debug port ............................................. 1352
- 41.4.2 SW debug port ............................................... 1355
- 41.4.3 Debug port registers ........................................ 1356
- 41.4.4 DP debug port identification register (DP_DPIDR) ...... 1357
- 41.4.5 DP abort register (DP_ABORTR) ............................ 1357
- 41.4.6 DP control and status register (DP_CTRL/STATR) ....... 1358
- 41.4.7 DP data link control register (DP_DLCR) ................. 1360
- 41.4.8 DP target identification register (DP_TARGETIDR) ....... 1360
- 41.4.9 DP data link protocol identification register (DP_DLPIDR) 1361
- 41.4.10 DP resend register (DP_RESENDR) .......................... 1361
- 41.4.11 DP access port select register (DP_SELECTR) .......... 1362
- 41.4.12 DP read buffer register (DP_BUFFR) ...................... 1362
- 41.4.13 DP target selection register (DP_TARGETSELR) ......... 1363
- 41.4.14 Debug port register map and reset values ............... 1364

#### 41.5 Access ports .................................................. 1365

- 41.5.1 AP control/status word register (AP_CSWR) ............ 1368
- 41.5.2 AP transfer address register (AP_TAR) ................... 1369
- 41.5.3 AP data read/write register (AP_DWR) .................... 1369
- 41.5.4 AP banked data registers (AP_BD0-3R) .................... 1369
- 41.5.5 AP base address register (AP_BASER) .................... 1370
41.5.6 AP identification register (AP_IDR) .......................... 1370
41.5.7 Access port register map and reset values ....................... 1372
41.6 Cross trigger interface (CTI) and matrix (CTM) ................. 1373
41.7 Cross trigger interface registers ................................ 1377
  41.7.1 CTI control register (CTI_CONTROLR) ....................... 1377
  41.7.2 CTI trigger acknowledge register (CTI_INTACKR) .......... 1377
  41.7.3 CTI application trigger set register (CTI_APPSETR) ..... 1377
  41.7.4 CTI application trigger clear register (CTI_APPCLEAR) .... 1378
  41.7.5 CTI application pulse register (CTI_APPPULSER) ........ 1379
  41.7.6 CTI trigger In x enable register (CTI_INENRx) ............ 1379
  41.7.7 CTI trigger out x enable register (CTI_OUTENRx) ....... 1380
  41.7.8 CTI trigger in status register (CTI_TRGINSTSR) .......... 1380
  41.7.9 CTI trigger out status register (CTI_TRGOSTSR) .......... 1381
  41.7.10 CTI channel in status register (CTI_CHINSTSR) ......... 1381
  41.7.11 CTI channel out status register (CTI_CHOUTSTSR) ....... 1381
  41.7.12 CTI channel gate register (CTI_GATER) .................. 1382
  41.7.13 CTI claim tag set register (CTI_CLAIMSETR) .......... 1382
  41.7.14 CTI claim tag clear register (CTI_CLAIMCLR) ........... 1383
  41.7.15 CTI lock access register (CTI_LAR) ..................... 1383
  41.7.16 CTI lock status register (CTI_LSR) ................. 1384
  41.7.17 CTI authentication status register (CTI_AUTHSTATR) .... 1384
  41.7.18 CTI device configuration register (CTI_DEVIDR) ....... 1385
  41.7.19 CTI device type identifier register (CTI_DEVTYPEPER) .. 1385
  41.7.20 CTI CoreSight peripheral identity register 4 (CTI_PIDR4) 1386
  41.7.21 CTI CoreSight peripheral identity register 0 (CTI_PIDR0) 1386
  41.7.22 CTI CoreSight peripheral identity register 1 (CTI_PIDR1) 1386
  41.7.23 CTI CoreSight peripheral identity register 2 (CTI_PIDR2) 1387
  41.7.24 CTI CoreSight peripheral identity register 3 (CTI_PIDR3) 1387
  41.7.25 CTI CoreSight component identity register 0 (CTI_CIDR0) 1388
  41.7.26 CTI CoreSight peripheral identity register 1 (CTI_CIDR1) 1388
  41.7.27 CTI CoreSight component identity register 2 (CTI_CIDR2) 1389
  41.7.28 CTI CoreSight component identity register 3 (CTI_CIDR3) 1389
  41.7.29 CTI register map and reset values ...................... 1390
41.8 Microcontroller debug unit (DBGMCU) ........................... 1393
  41.8.1 DBGMCU identity code register (DBGMCU_IDCODE) .... 1393
  41.8.2 DBGMCU configuration register (DBGMCU_CR) .......... 1393
41.8.3 DBGMCU CPU1 APB1 peripheral freeze register 1
(DBGMCU_APB1FZR1) ........................................... 1394
41.8.4 DBGMCU CPU2 APB1 peripheral freeze register 1
(DBGMCU_C2APB1FZR1) ....................................... 1395
41.8.5 DBGMCU CPU1 APB1 peripheral freeze register 2
(DBGMCU_APB1FZR2) ........................................... 1396
41.8.6 DBGMCU CPU2 APB1 peripheral freeze register 2
(DBGMCU_C2APB1FZR2) ....................................... 1397
41.8.7 DBGMCU CPU1 APB2 peripheral freeze register
(DBGMCU_APB2FZR) ............................................. 1397
41.8.8 DBGMCU CPU2 APB2 peripheral freeze register
(DBGMCU_C2APB2FZR) ........................................ 1398
41.8.9 DBGMCU register map and reset values ................. 1400

41.9 CPU2 ROM tables ........................................... 1402
41.9.1 CPU2 ROM1 memory type register (C2ROM1_MEMTYPEPER) ........... 1404
41.9.2 CPU2 ROM1 CoreSight peripheral identity register 4
(C2ROM1_PIDR4) ............................................. 1404
41.9.3 CPU2 ROM1 CoreSight peripheral identity register 0
(C2ROM1_PIDR0) ............................................. 1404
41.9.4 CPU2 ROM1 CoreSight peripheral identity register 1
(C2ROM1_PIDR1) ............................................. 1405
41.9.5 CPU2 ROM1 CoreSight peripheral identity register 2
(C2ROM1_PIDR2) ............................................. 1405
41.9.6 CPU2 ROM1 CoreSight peripheral identity register 3
(C2ROM1_PIDR3) ............................................. 1406
41.9.7 CPU2 ROM1 CoreSight component identity register 0
(C2ROM1_CIDR0) ............................................. 1406
41.9.8 CPU2 ROM1 CoreSight peripheral identity register 1
(C2ROM1_CIDR1) ............................................. 1407
41.9.9 CPU2 ROM1 CoreSight component identity register 2
(C2ROM1_CIDR2) ............................................. 1407
41.9.10 CPU2 ROM1 CoreSight component identity register 3
(C2ROM1_CIDR3) ............................................. 1407
41.9.11 CPU2 processor ROM table registers and reset values ........... 1408
41.9.12 CPU2 ROM2 memory type register (C2ROM2_MEMTYPEPER) ........ 1410
41.9.13 CPU2 ROM2 CoreSight peripheral identity register 4
(C2ROM2_PIDR4) ............................................. 1410
41.9.14 CPU2 ROM2 CoreSight peripheral identity register 0
(C2ROM2_PIDR0) ............................................. 1410
41.9.15 CPU2 ROM2 CoreSight peripheral identity register 1
(C2ROM2_PIDR1) ............................................. 1411
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>41.9.16</td>
<td>CPU2 ROM2 CoreSight peripheral identity register 2 (C2ROM2_PIDR2)</td>
</tr>
<tr>
<td>41.9.17</td>
<td>CPU2 ROM2 CoreSight peripheral identity register 3 (C2ROM2_PIDR3)</td>
</tr>
<tr>
<td>41.9.18</td>
<td>CPU2 ROM2 CoreSight component identity register 0 (C2ROM2_CIDR0)</td>
</tr>
<tr>
<td>41.9.19</td>
<td>CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_CIDR1)</td>
</tr>
<tr>
<td>41.9.20</td>
<td>CPU2 ROM2 CoreSight component identity register 2 (C2ROM2_CIDR2)</td>
</tr>
<tr>
<td>41.9.21</td>
<td>CPU2 ROM2 CoreSight component identity register 3 (C2ROM2_CIDR3)</td>
</tr>
<tr>
<td>41.9.22</td>
<td>CPU2 ROM table register map and reset values</td>
</tr>
<tr>
<td>41.10</td>
<td>CPU2 data watchpoint and trace unit (DWT)</td>
</tr>
<tr>
<td>41.10.1</td>
<td>DWT control register (DWT_CTRLR)</td>
</tr>
<tr>
<td>41.10.2</td>
<td>DWT cycle count register (DWT_CYCCNTR)</td>
</tr>
<tr>
<td>41.10.3</td>
<td>DWT CPI count register (DWT_CPICNTR)</td>
</tr>
<tr>
<td>41.10.4</td>
<td>DWT exception count register (DWT_EXCCNTR)</td>
</tr>
<tr>
<td>41.10.5</td>
<td>DWT sleep count register (DWT_SLPCNTR)</td>
</tr>
<tr>
<td>41.10.6</td>
<td>DWT LSU count register (DWT_LSUCNTR)</td>
</tr>
<tr>
<td>41.10.7</td>
<td>DWT fold count register (DWT_FOLDCNTR)</td>
</tr>
<tr>
<td>41.10.8</td>
<td>DWT program counter sample register (DWT_PCSR)</td>
</tr>
<tr>
<td>41.10.9</td>
<td>DWT comparator register x (DWT_COMPxR)</td>
</tr>
<tr>
<td>41.10.10</td>
<td>DWT mask register x (DWT_MASKxR)</td>
</tr>
<tr>
<td>41.10.11</td>
<td>DWT function register x (DWT_FUNCTxR)</td>
</tr>
<tr>
<td>41.10.12</td>
<td>DWT CoreSight peripheral identity register 4 (DWT_PIDR4)</td>
</tr>
<tr>
<td>41.10.13</td>
<td>DWT CoreSight peripheral identity register 0 (DWT_PIDR0)</td>
</tr>
<tr>
<td>41.10.14</td>
<td>DWT CoreSight peripheral identity register 1 (DWT_PIDR1)</td>
</tr>
<tr>
<td>41.10.15</td>
<td>DWT CoreSight peripheral identity register 2 (DWT_PIDR2)</td>
</tr>
<tr>
<td>41.10.16</td>
<td>DWT CoreSight peripheral identity register 3 (DWT_PIDR3)</td>
</tr>
<tr>
<td>41.10.17</td>
<td>DWT CoreSight component identity register 0 (DWT_CIDR0)</td>
</tr>
<tr>
<td>41.10.18</td>
<td>DWT CoreSight peripheral identity register 1 (DWT_CIDR1)</td>
</tr>
<tr>
<td>41.10.19</td>
<td>DWT CoreSight component identity register 2 (DWT_CIDR2)</td>
</tr>
<tr>
<td>41.10.20</td>
<td>DWT CoreSight component identity register 3 (DWT_CIDR3)</td>
</tr>
<tr>
<td>41.10.21</td>
<td>CPU2 DWT registers</td>
</tr>
<tr>
<td>41.11</td>
<td>CPU2 breakpoint unit (PBU)</td>
</tr>
<tr>
<td>41.11.1</td>
<td>BPU control register (BPU_CTRLR)</td>
</tr>
<tr>
<td>41.11.2</td>
<td>BPU remap register (BPU_REMAPR)</td>
</tr>
<tr>
<td>41.11.3</td>
<td>BPU comparator registers (BPU_COMPxR)</td>
</tr>
</tbody>
</table>
41.11.4 BPU CoreSight peripheral identity register 4 (BPU_PIDR4) ......... 1431
41.11.5 BPU CoreSight peripheral identity register 0 (BPU_PIDR0) ........ 1432
41.11.6 BPU CoreSight peripheral identity register 1 (BPU_PIDR1) ........ 1432
41.11.7 BPU CoreSight peripheral identity register 2 (BPU_PIDR2) ........ 1432
41.11.8 BPU CoreSight peripheral identity register 3 (BPU_PIDR3) ........ 1433
41.11.9 BPU CoreSight component identity register 0 (BPU_CIDR0) ....... 1433
41.11.10 BPU CoreSight peripheral identity register 1 (BPU_CIDR1) ....... 1434
41.11.11 BPU CoreSight component identity register 2 (BPU_CIDR2) ....... 1434
41.11.12 BPU CoreSight component identity register 3 (BPU_CIDR3) ....... 1435
41.11.13 CPU2 BPU register map and reset values ........................... 1436

41.12 CPU2 cross trigger interface (CTI) ................................. 1437

41.13 CPU1 ROM table ..................................................... 1437
41.13.1 CPU1 ROM memory type register (C1ROM_MEMTYPERM) .............. 1438
41.13.2 CPU1 ROM CoreSight peripheral identity register 4 (C1ROM_PIDR4) .................................................. 1439
41.13.3 CPU1 ROM CoreSight peripheral identity register 0 (C1ROM_PIDR0) .................................................. 1439
41.13.4 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_PIDR1) .................................................. 1440
41.13.5 CPU1 ROM CoreSight peripheral identity register 2 (C1ROM_PIDR2) .................................................. 1440
41.13.6 CPU1 ROM CoreSight peripheral identity register 3 (C1ROM_PIDR3) .................................................. 1441
41.13.7 CPU1 ROM CoreSight component identity register 0 (C1ROM_CIDR0) .................................................. 1441
41.13.8 CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_CIDR1) .................................................. 1441
41.13.9 CPU1 ROM CoreSight component identity register 2 (C1ROM_CIDR2) .................................................. 1442
41.13.10 CPU1 ROM CoreSight component identity register 3 (C1ROM_CIDR3) .................................................. 1442
41.13.11 CPU1 ROM table register map and reset values ...................... 1444

41.14 CPU1 data watchpoint and trace unit (DWT) .......................... 1445
41.14.1 DWT control register (DWT_CTRLR) ................................ 1445
41.14.2 DWT cycle count register (DWT_CYCCNTR) ........................ 1447
41.14.3 DWT CPI count register (DWT_CPICNTR) ........................... 1447
41.14.4 DWT exception count register (DWT_EXCCNTR) .................... 1448
41.14.5 DWT sleep count register (DWT_SLPCNTR) ........................ 1448
41.14.6 DWT LSU count register (DWT_LSUCNTR) ........................ 1449
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>41.14.7</td>
<td>DWT fold count register (DWT_FOLDCNTR)</td>
</tr>
<tr>
<td>41.14.8</td>
<td>DWT program counter sample register (DWT_PCSR)</td>
</tr>
<tr>
<td>41.14.9</td>
<td>DWT comparator register x (DWT_COMPxR)</td>
</tr>
<tr>
<td>41.14.10</td>
<td>DWT mask register x (DWT_MASKxR)</td>
</tr>
<tr>
<td>41.14.11</td>
<td>DWT function register x (DWT_FUNCTxR)</td>
</tr>
<tr>
<td>41.14.12</td>
<td>DWT CoreSight peripheral identity register 4 (DWT_PIDR4)</td>
</tr>
<tr>
<td>41.14.13</td>
<td>DWT CoreSight peripheral identity register 0 (DWT_PIDR0)</td>
</tr>
<tr>
<td>41.14.14</td>
<td>DWT CoreSight peripheral identity register 1 (DWT_PIDR1)</td>
</tr>
<tr>
<td>41.14.15</td>
<td>DWT CoreSight peripheral identity register 2 (DWT_PIDR2)</td>
</tr>
<tr>
<td>41.14.16</td>
<td>DWT CoreSight peripheral identity register 3 (DWT_PIDR3)</td>
</tr>
<tr>
<td>41.14.17</td>
<td>DWT CoreSight component identity register 0 (DWT_CIDR0)</td>
</tr>
<tr>
<td>41.14.18</td>
<td>DWT CoreSight peripheral identity register 1 (DWT_CIDR1)</td>
</tr>
<tr>
<td>41.14.19</td>
<td>DWT CoreSight component identity register 2 (DWT_CIDR2)</td>
</tr>
<tr>
<td>41.14.20</td>
<td>DWT CoreSight component identity register 3 (DWT_CIDR3)</td>
</tr>
<tr>
<td>41.14.21</td>
<td>CPU1 DWT register map and reset values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>41.15</td>
<td>CPU1 instrumentation trace macrocell (ITM)</td>
</tr>
<tr>
<td>41.15.1</td>
<td>ITM stimulus register x (ITM_STIMRx)</td>
</tr>
<tr>
<td>41.15.2</td>
<td>ITM trace enable register (ITM_TER)</td>
</tr>
<tr>
<td>41.15.3</td>
<td>ITM trace privilege register (ITM_TPR)</td>
</tr>
<tr>
<td>41.15.4</td>
<td>ITM trace control register (ITM_TCR)</td>
</tr>
<tr>
<td>41.15.5</td>
<td>ITM CoreSight peripheral identity register 4 (ITM_PIDR4)</td>
</tr>
<tr>
<td>41.15.6</td>
<td>ITM CoreSight peripheral identity register 0 (ITM_PIDR0)</td>
</tr>
<tr>
<td>41.15.7</td>
<td>ITM CoreSight peripheral identity register 1 (ITM_PIDR1)</td>
</tr>
<tr>
<td>41.15.8</td>
<td>ITM CoreSight peripheral identity register 2 (ITM_PIDR2)</td>
</tr>
<tr>
<td>41.15.9</td>
<td>ITM CoreSight peripheral identity register 3 (ITM_PIDR3)</td>
</tr>
<tr>
<td>41.15.10</td>
<td>ITM CoreSight component identity register 0 (ITM_CIDR0)</td>
</tr>
<tr>
<td>41.15.11</td>
<td>ITM CoreSight peripheral identity register 1 (ITM_CIDR1)</td>
</tr>
<tr>
<td>41.15.12</td>
<td>ITM CoreSight component identity register 2 (ITM_CIDR2)</td>
</tr>
<tr>
<td>41.15.13</td>
<td>ITM CoreSight component identity register 3 (ITM_CIDR3)</td>
</tr>
<tr>
<td>41.15.14</td>
<td>ITM register map and reset values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>41.16</td>
<td>CPU1 breakpoint unit (FPB)</td>
</tr>
<tr>
<td>41.16.1</td>
<td>FPB control register (FPB_CTRLR)</td>
</tr>
<tr>
<td>41.16.2</td>
<td>FPB remap register (FPB_REMAPR)</td>
</tr>
<tr>
<td>41.16.3</td>
<td>FPB comparator registers (FPB_COMPxR)</td>
</tr>
<tr>
<td>41.16.4</td>
<td>FPB CoreSight peripheral identity register 4 (FPB_PIDR4)</td>
</tr>
<tr>
<td>41.16.5</td>
<td>FPB CoreSight peripheral identity register 0 (FPB_PIDR0)</td>
</tr>
<tr>
<td>41.16.6</td>
<td>FPB CoreSight peripheral identity register 1 (FPB_PIDR1)</td>
</tr>
</tbody>
</table>
41.16.7 FPB CoreSight peripheral identity register 2 (FPB_PIDR2) .......... 1469
41.16.8 FPB CoreSight peripheral identity register 3 (FPB_PIDR3) .......... 1469
41.16.9 FPB CoreSight component identity register 0 (FPB_CIDR0) .......... 1470
41.16.10 FPB CoreSight peripheral identity register 1 (FPB_CIDR1) ........ 1470
41.16.11 FPB CoreSight component identity register 2 (FPB_CIDR2) ........ 1470
41.16.12 FPB CoreSight component identity register 3 (FPB_CIDR3) ........ 1471
41.16.13 FPB register map and reset values ............................ 1472

41.17 CPU1 Embedded trace macrocell (ETM™), only available on STM32WB55xx .............................................. 1473
41.17.1 ETM control register (ETM_CR) ...................... 1473
41.17.2 ETM configuration code register (ETM_CCR) .............. 1474
41.17.3 ETM trigger register (ETM_TRIGGER) ................. 1475
41.17.4 ETM status register (ETM_SR) ......................... 1476
41.17.5 ETM status register (ETMSCR) ....................... 1476
41.17.6 ETM trace enable event register (ETM_TEEVR) .......... 1477
41.17.7 ETM trace enable control 1 register (ETM_TECR1) ......... 1478
41.17.8 ETM FIFO FULL level register (ETM_FFLR) ............ 1478
41.17.9 ETM counter reload value 1 register (ETM_CNTRLDVR1) ..... 1479
41.17.10 ETM synchronization frequency register (ETM_SYNCFR) .... 1479
41.17.11 ETM ID register (ETM_IDR) ............................ 1480
41.17.12 ETM configuration code extension register (ETM_CCER) .... 1480
41.17.13 ETM trace enable start/stop EmbeddedICE control register (ETM_TESSEICR) .............................................. 1481
41.17.14 ETM timestamp event register (ETM_TSEVR) .......... 1482
41.17.15 ETM trace ID register (ETM_TRACEIDR) ................ 1483
41.17.16 ETM ID register 2(ETM_IDR2) .......................... 1483
41.17.17 ETM device power down status register 2(ETM_PDSR) ....... 1483
41.17.18 ETM claim tag set register (ETM_CLAIMSETR) ........ 1484
41.17.19 ETM claim tag clear register (ETM_CLAIMCLR) .......... 1484
41.17.20 ETM lock access register (ETM_LAR) .................. 1485
41.17.21 ETM lock status register (ETM_LSR) .................. 1485
41.17.22 ETM authentication status register (ETM_AUTHSTATR) .... 1486
41.17.23 ETM CoreSight device identity register (ETM_DEVTYPEPR) ...... 1486
41.17.24 ETM CoreSight peripheral identity register 4 (ETM_PIDR4) .... 1487
41.17.25 ETM CoreSight peripheral identity register 0 (ETM_PIDR0) .... 1487
41.17.26 ETM CoreSight peripheral identity register 1 (ETM_PIDR1) .... 1487
41.17.27 ETM CoreSight peripheral identity register 2 (ETM_PIDR2) .... 1488
41.17.28 ETM CoreSight peripheral identity register 3 (ETM_PIDR3) .................. 1488
41.17.29 ETM CoreSight component identity register 0 (ETM_CIDR0) .................. 1489
41.17.30 ETM CoreSight peripheral identity register 1 (ETM_CIDR1) .................. 1489
41.17.31 ETM CoreSight component identity register 2 (ETM_CIDR2) .................. 1490
41.17.32 ETM CoreSight component identity register 3 (ETM_CIDR3) .................. 1490
41.17.33 ETM register map and reset values .............................................. 1491

41.18 CPU1 trace port interface unit (TPIU) ............................................. 1494
  41.18.1 TPIU supported port size register (TPIU_SSPSR) .............................. 1494
  41.18.2 TPIU current port size register (TPIU_CSPSR) ................................ 1495
  41.18.3 TPIU asynchronous clock presecaler register (TPIU_ACPR) .................. 1495
  41.18.4 TPIU selected pin protocol register (TPIU_SPPR) ............................ 1495
  41.18.5 TPIU formatter and flush status register (TPIU_FFSR) ...................... 1496
  41.18.6 TPIU formatter and flush control register (TPIU_FFCR) ..................... 1496
  41.18.7 TPIU formatter synchronization counter register (TPIU_FSCR) ............. 1497
  41.18.8 TPIU claim tag set register (TPIU_CLAIMSETR) .............................. 1497
  41.18.9 TPIU claim tag clear register (TPIU_CLAIMCLR) .............................. 1498
  41.18.10 TPIU device configuration register (TPIU_DEVIDR) ......................... 1498
  41.18.11 TPIU device type identifier register (TPIU_DEVTYPEPR) ................... 1499
  41.18.12 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4) ............. 1499
  41.18.13 TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0) ............. 1500
  41.18.14 TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1) ............. 1500
  41.18.15 TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2) ............. 1501
  41.18.16 TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3) ............. 1501
  41.18.17 TPIU CoreSight component identity register 0 (TPIU_CIDR0) ............. 1502
  41.18.18 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1) ............. 1502
  41.18.19 TPIU CoreSight component identity register 2 (TPIU_CIDR2) ............. 1502
  41.18.20 TPIU CoreSight component identity register 3 (TPIU_CIDR3) ............. 1503
  41.18.21 CPU1 TPIU register map and reset values .................................... 1504

41.19 CPU1 cross trigger interface (CTI) .............................................. 1506

41.20 References ...................................................................................... 1506

42 Device electronic signature ................................................................. 1507
  42.1 Unique device ID register (96 bits) .................................................. 1507
  42.2 Memory size data register .............................................................. 1508
    42.2.1 Flash size data register ......................................................... 1508
  42.3 Package data register ..................................................................... 1508
42.4 Part number codification register . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1509

43 Important security notice . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1510

44 Revision history . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1511
# List of tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1.</td>
<td>Memory map and peripheral register boundary addresses</td>
</tr>
<tr>
<td>Table 2.</td>
<td>Boot modes</td>
</tr>
<tr>
<td>Table 3.</td>
<td>Flash memory - Single bank organization</td>
</tr>
<tr>
<td>Table 4.</td>
<td>Number of wait states vs. Flash memory clock (HCLK4) frequency</td>
</tr>
<tr>
<td>Table 5.</td>
<td>Page erase overview</td>
</tr>
<tr>
<td>Table 6.</td>
<td>Mass erase overview</td>
</tr>
<tr>
<td>Table 7.</td>
<td>Errors in page-based row programming</td>
</tr>
<tr>
<td>Table 8.</td>
<td>Option bytes format</td>
</tr>
<tr>
<td>Table 9.</td>
<td>Option bytes organization</td>
</tr>
<tr>
<td>Table 10.</td>
<td>Option loading control</td>
</tr>
<tr>
<td>Table 11.</td>
<td>UID64 organization</td>
</tr>
<tr>
<td>Table 12.</td>
<td>Flash memory read protection status</td>
</tr>
<tr>
<td>Table 13.</td>
<td>RDP regression from Level 1 to Level 0 and memory erase</td>
</tr>
<tr>
<td>Table 14.</td>
<td>Access status vs. protection level and execution modes</td>
</tr>
<tr>
<td>Table 15.</td>
<td>Flash memory interrupt requests</td>
</tr>
<tr>
<td>Table 16.</td>
<td>Flash interface register map and reset values</td>
</tr>
<tr>
<td>Table 17.</td>
<td>CRC internal input/output signals</td>
</tr>
<tr>
<td>Table 18.</td>
<td>CRC register map and reset values</td>
</tr>
<tr>
<td>Table 19.</td>
<td>Supply configuration control</td>
</tr>
<tr>
<td>Table 20.</td>
<td>PVM features</td>
</tr>
<tr>
<td>Table 21.</td>
<td>Sub-system low power wake-up sources</td>
</tr>
<tr>
<td>Table 22.</td>
<td>Low-power mode summary</td>
</tr>
<tr>
<td>Table 23.</td>
<td>Functionalities depending on system operating mode</td>
</tr>
<tr>
<td>Table 24.</td>
<td>Low-power run</td>
</tr>
<tr>
<td>Table 25.</td>
<td>CPU CSTOP wake-up vs. system operating mode</td>
</tr>
<tr>
<td>Table 26.</td>
<td>Sleep mode</td>
</tr>
<tr>
<td>Table 27.</td>
<td>Low-power sleep</td>
</tr>
<tr>
<td>Table 28.</td>
<td>Stop0 mode</td>
</tr>
<tr>
<td>Table 29.</td>
<td>Stop1 mode</td>
</tr>
<tr>
<td>Table 30.</td>
<td>Stop2 mode</td>
</tr>
<tr>
<td>Table 31.</td>
<td>Standby mode</td>
</tr>
<tr>
<td>Table 32.</td>
<td>Shutdown mode</td>
</tr>
<tr>
<td>Table 33.</td>
<td>PWR register map and reset values</td>
</tr>
<tr>
<td>Table 34.</td>
<td>Interconnect matrix implementation</td>
</tr>
<tr>
<td>Table 35.</td>
<td>Peripherals interconnect matrix</td>
</tr>
<tr>
<td>Table 36.</td>
<td>Maximum clock source frequency</td>
</tr>
<tr>
<td>Table 37.</td>
<td>SMPS step-down converter clock source selection and division</td>
</tr>
<tr>
<td>Table 38.</td>
<td>Peripheral clock enable</td>
</tr>
<tr>
<td>Table 39.</td>
<td>Single core Low power debug configurations</td>
</tr>
<tr>
<td>Table 40.</td>
<td>RCC register map and reset values</td>
</tr>
<tr>
<td>Table 41.</td>
<td>GPIO implementation</td>
</tr>
<tr>
<td>Table 42.</td>
<td>Port bit configuration table</td>
</tr>
<tr>
<td>Table 43.</td>
<td>GPIO register map and reset values</td>
</tr>
<tr>
<td>Table 44.</td>
<td>SYSCFG register map and reset values</td>
</tr>
<tr>
<td>Table 45.</td>
<td>DMA1 and DMA2 implementation</td>
</tr>
<tr>
<td>Table 46.</td>
<td>Programmable data width and endian behavior (when PINC = MINC = 1)</td>
</tr>
<tr>
<td>Table 47.</td>
<td>DMA interrupt requests</td>
</tr>
<tr>
<td>Table 48.</td>
<td>DMA register map and reset values</td>
</tr>
</tbody>
</table>
Table 51. DMAMUX implementation ................................................. 350
Table 52. DMAMUX instantiation .................................................. 350
Table 53. DMAMUX: assignment of multiplexer inputs to resources ..... 351
Table 54. DMAMUX: assignment of trigger inputs to resources .......... 351
Table 55. DMAMUX: assignment of synchronization inputs to resources 352
Table 56. DMAMUX signals ......................................................... 354
Table 57. DMAMUX interrupts ...................................................... 358
Table 58. DMAMUX register map and reset values ....................... 363
Table 59. NVIC implementation .................................................. 365
Table 60. CPU1 vector table ...................................................... 367
Table 61. CPU2 vector table ...................................................... 370
Table 62. Wakeup interrupt table ................................................ 372
Table 63. EXTI implementation ................................................... 374
Table 64. EXTI pin overview ........................................................ 375
Table 65. EVG pin overview ....................................................... 376
Table 66. EXTI event input configurations and register control ......... 377
Table 67. Masking functionality ................................................... 380
Table 68. EXTI register map sections ............................................. 381
Table 69. EXTI register map and reset values ............................... 391
Table 70. QUADSPI pins ............................................................. 394
Table 71. QUADSPI interrupt requests ......................................... 406
Table 72. QUADSPI register map and reset values ....................... 417
Table 73. ADC features ............................................................. 420
Table 74. ADC internal input/output signals ................................. 422
Table 75. ADC input/output pins .................................................. 422
Table 76. Configuring the trigger polarity for regular external triggers 438
Table 77. Configuring the trigger polarity for injected external triggers 438
Table 78. ADC1 - External triggers for regular channels ............... 439
Table 79. ADC1 - External trigger for injected channels ............... 439
Table 80. TSAR timings depending on resolution ............................ 451
Table 81. Offset computation versus data resolution ...................... 454
Table 82. Analog watchdog channel selection ............................... 464
Table 83. Analog watchdog 1 comparison ...................................... 465
Table 84. Analog watchdog 2 and 3 comparison ............................ 465
Table 85. Maximum output results versus N and M (gray cells indicate truncation) 469
Table 86. Effect of low-power modes on the ADC .......................... 477
Table 87. ADC interrupts ........................................................... 478
Table 88. ADC register map and reset values .................................. 507
Table 89. ADC register map and reset values (master and slave ADC common registers) offset = 0x300 509
Table 90. VREFBUF implementation ............................................. 510
Table 91. VREF buffer modes ..................................................... 510
Table 92. VREFBUF trimming data .............................................. 511
Table 93. VREFBUF register map and reset values ....................... 512
Table 94. COMP1 input plus assignment ...................................... 515
Table 95. COMP1 input minus assignment .................................... 516
Table 96. COMP2 input plus assignment ...................................... 516
Table 97. COMP2 input minus assignment .................................... 516
Table 98. Comparator behavior in the low power modes ................. 520
Table 99. Interrupt control bits .................................................. 520
Table 100. COMP register map and reset values ........................... 526
Table 101. Example of frame rate calculation ................................. 530
Table 153. ECDSA verification average computation times
Table 152. ECDSA signature average computation times
Table 150. Modular exponentiation computation times
Table 149. Family of supported curves for ECC operations
Table 148. ECDSA verification (outputs)
Table 147. ECDSA verification (inputs)
Table 145. ECDSA sign - Outputs
Table 144. ECDSA sign - Inputs
Table 143. ECC Fp scalar multiplication (Fast Mode)
Table 140. CRT exponentiation
Table 139. Arithmetic comparison
Table 138. Arithmetic multiplication
Table 137. Arithmetic subtraction
Table 136. Arithmetic addition
Table 135. Modular reduction
Table 134. Modular inversion
Table 133. Modular exponentiation (fast mode)
Table 132. Modular exponentiation (normal mode)
Table 131. Montgomery multiplication
Table 130. Modular subtraction
Table 129. Modular addition
Table 128. Montgomery parameter computation
Table 127. PKA prime field (Fp) elliptic curve functions list
Table 125. Internal input/output signals
Table 124. AES register map and reset values
Table 123. Processing latency for GCM and CCM (in clock cycles)
Table 122. Processing latency for ECB, CBC and CTR
Table 121. Key endianness in AES_KEYRx registers (128- or 256-bit key length)
Table 120. AES interrupt requests
Table 119. Initialization of AES_IVRx registers in CCM mode
Table 118. Initialization of AES_IVRx registers in GCM mode
Table 117. GCM last block definition
Table 116. CTR mode initialization vector definition
Table 115. AES register map and reset values
Table 114. RNG register map and reset map
Table 113. RNG interrupt requests
Table 112. RNG internal input/output signals
Table 111. TSC register map and reset values
Table 110. Interrupt control bits
Table 109. Effect of low-power modes on TSC
Table 108. I/O state depending on its mode and IODEF bit value
Table 107. Spread spectrum deviation versus AHB clock frequency
Table 106. Acquisition sequence summary
Table 105. LCD register map and reset values
Table 104. LCD interrupt requests
Table 103. Remapping capability
Table 102. Blink frequency
Table 101. Processing latency for AES
Table 100. Effect of power modes on TSC
Table 99. I/O state depending on its mode and TSC_MODE bit value
Table 98. Spread spectrum deviation versus AHB clock frequency
Table 97. Acquisition sequence summary
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>154</td>
<td>Point on elliptic curve $F_p$ check average computation times</td>
<td>659</td>
</tr>
<tr>
<td>155</td>
<td>Montgomery parameters average computation times</td>
<td>659</td>
</tr>
<tr>
<td>156</td>
<td>PKA interrupt requests</td>
<td>663</td>
</tr>
<tr>
<td>157</td>
<td>PKA register map and reset values</td>
<td>665</td>
</tr>
<tr>
<td>158</td>
<td>Behavior of timer outputs versus BRK/BRK2 inputs</td>
<td>706</td>
</tr>
<tr>
<td>159</td>
<td>Break protection disarming conditions</td>
<td>708</td>
</tr>
<tr>
<td>160</td>
<td>Counting direction versus encoder signals</td>
<td>714</td>
</tr>
<tr>
<td>161</td>
<td>TIM1 internal trigger connection</td>
<td>731</td>
</tr>
<tr>
<td>162</td>
<td>Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)</td>
<td>745</td>
</tr>
<tr>
<td>163</td>
<td>TIM1 register map and reset values</td>
<td>762</td>
</tr>
<tr>
<td>164</td>
<td>Counting direction versus encoder signals</td>
<td>798</td>
</tr>
<tr>
<td>165</td>
<td>TIM2 internal trigger connection</td>
<td>816</td>
</tr>
<tr>
<td>166</td>
<td>Output control bit for standard OCx channels.</td>
<td>827</td>
</tr>
<tr>
<td>167</td>
<td>TIM2 register map and reset values</td>
<td>834</td>
</tr>
<tr>
<td>168</td>
<td>Break protection disarming conditions</td>
<td>859</td>
</tr>
<tr>
<td>169</td>
<td>Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)</td>
<td>876</td>
</tr>
<tr>
<td>170</td>
<td>TIM16/TIM17 register map and reset values</td>
<td>887</td>
</tr>
<tr>
<td>171</td>
<td>LPTIM features</td>
<td>890</td>
</tr>
<tr>
<td>172</td>
<td>LPTIM implementation</td>
<td>890</td>
</tr>
<tr>
<td>173</td>
<td>LPTIM1 external trigger connection</td>
<td>891</td>
</tr>
<tr>
<td>174</td>
<td>LPTIM2 external trigger connection</td>
<td>892</td>
</tr>
<tr>
<td>175</td>
<td>Prescaler division ratios</td>
<td>893</td>
</tr>
<tr>
<td>176</td>
<td>Encoder counting scenarios</td>
<td>900</td>
</tr>
<tr>
<td>177</td>
<td>Effect of low-power modes on the LPTIM</td>
<td>901</td>
</tr>
<tr>
<td>178</td>
<td>Interrupt events</td>
<td>902</td>
</tr>
<tr>
<td>179</td>
<td>LPTIM register map and reset values</td>
<td>913</td>
</tr>
<tr>
<td>180</td>
<td>RTC implementation</td>
<td>916</td>
</tr>
<tr>
<td>181</td>
<td>Effect of low-power modes on RTC</td>
<td>930</td>
</tr>
<tr>
<td>182</td>
<td>Interrupt control bits</td>
<td>930</td>
</tr>
<tr>
<td>183</td>
<td>RTC register map and reset values</td>
<td>955</td>
</tr>
<tr>
<td>184</td>
<td>IWDG register map and reset values</td>
<td>965</td>
</tr>
<tr>
<td>185</td>
<td>WWWDG register map and reset values</td>
<td>971</td>
</tr>
<tr>
<td>186</td>
<td>I2C implementation</td>
<td>973</td>
</tr>
<tr>
<td>187</td>
<td>I2C input/output pins</td>
<td>974</td>
</tr>
<tr>
<td>188</td>
<td>I2C internal input/output signals</td>
<td>975</td>
</tr>
<tr>
<td>189</td>
<td>Comparison of analog and digital filters</td>
<td>977</td>
</tr>
<tr>
<td>190</td>
<td>$I^C$-SMBus specification data setup and hold times</td>
<td>979</td>
</tr>
<tr>
<td>191</td>
<td>I2C configuration.</td>
<td>983</td>
</tr>
<tr>
<td>192</td>
<td>$I^C$-SMBus specification clock timings</td>
<td>994</td>
</tr>
<tr>
<td>193</td>
<td>Timing settings for $f_{2CCLK}$ of 8 MHz.</td>
<td>1004</td>
</tr>
<tr>
<td>194</td>
<td>Timing settings for $f_{2CCLK}$ of 16 MHz.</td>
<td>1004</td>
</tr>
<tr>
<td>195</td>
<td>SMBus timeout specifications.</td>
<td>1006</td>
</tr>
<tr>
<td>196</td>
<td>SMBus with PEC configuration</td>
<td>1008</td>
</tr>
<tr>
<td>197</td>
<td>TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms.</td>
<td>1009</td>
</tr>
<tr>
<td>198</td>
<td>TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms</td>
<td>1009</td>
</tr>
<tr>
<td>199</td>
<td>TIMEOUTA[11:0] for maximum $t_{IDLE}$ of 50 $\mu$s</td>
<td>1009</td>
</tr>
<tr>
<td>200</td>
<td>Effect of low-power modes to I2C.</td>
<td>1019</td>
</tr>
<tr>
<td>201</td>
<td>I2C interrupt requests.</td>
<td>1019</td>
</tr>
<tr>
<td>202</td>
<td>I2C register map and reset values</td>
<td>1034</td>
</tr>
<tr>
<td>203</td>
<td>USART / LPUART features</td>
<td>1038</td>
</tr>
<tr>
<td>204</td>
<td>USART/UART input/output pins</td>
<td>1041</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>205</td>
<td>USART internal input/output signals</td>
<td>1041</td>
</tr>
<tr>
<td>206</td>
<td>Noise detection from sampled data</td>
<td>1053</td>
</tr>
<tr>
<td>207</td>
<td>Tolerance of the USART receiver when BRR[3:0] = 0000</td>
<td>1056</td>
</tr>
<tr>
<td>208</td>
<td>Tolerance of the USART receiver when BRR[3:0] is different from 0000</td>
<td>1057</td>
</tr>
<tr>
<td>209</td>
<td>USART frame formats</td>
<td>1062</td>
</tr>
<tr>
<td>210</td>
<td>Effect of low-power modes on the USART</td>
<td>1085</td>
</tr>
<tr>
<td>211</td>
<td>USART interrupt requests</td>
<td>1086</td>
</tr>
<tr>
<td>212</td>
<td>USART register map and reset values</td>
<td>1121</td>
</tr>
<tr>
<td>213</td>
<td>USART / LPUART features</td>
<td>1125</td>
</tr>
<tr>
<td>214</td>
<td>LPUART input/output pins</td>
<td>1127</td>
</tr>
<tr>
<td>215</td>
<td>LPUART internal input/output signals</td>
<td>1127</td>
</tr>
<tr>
<td>216</td>
<td>Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32.768 kHz</td>
<td>1137</td>
</tr>
<tr>
<td>217</td>
<td>Error calculation for programmed baud rates at fCK = 100 MHz</td>
<td>1138</td>
</tr>
<tr>
<td>218</td>
<td>Tolerance of the LPUART receiver</td>
<td>1139</td>
</tr>
<tr>
<td>219</td>
<td>Effect of low-power modes on the LPUART</td>
<td>1150</td>
</tr>
<tr>
<td>220</td>
<td>LPUART interrupt requests</td>
<td>1151</td>
</tr>
<tr>
<td>221</td>
<td>LPUART register map and reset values</td>
<td>1175</td>
</tr>
<tr>
<td>222</td>
<td>SPI implementation</td>
<td>1178</td>
</tr>
<tr>
<td>223</td>
<td>SPI interrupt requests</td>
<td>1202</td>
</tr>
<tr>
<td>224</td>
<td>SPI register map and reset values</td>
<td>1211</td>
</tr>
<tr>
<td>225</td>
<td>STM32WB55xx SAI features</td>
<td>1213</td>
</tr>
<tr>
<td>226</td>
<td>SAI internal input/output signals</td>
<td>1215</td>
</tr>
<tr>
<td>227</td>
<td>SAI input/output pins</td>
<td>1215</td>
</tr>
<tr>
<td>228</td>
<td>MCLK_x activation conditions</td>
<td>1222</td>
</tr>
<tr>
<td>229</td>
<td>Clock generator programming examples</td>
<td>1225</td>
</tr>
<tr>
<td>230</td>
<td>TDM settings</td>
<td>1232</td>
</tr>
<tr>
<td>231</td>
<td>TDM frame configuration examples</td>
<td>1234</td>
</tr>
<tr>
<td>232</td>
<td>SOPD pattern</td>
<td>1237</td>
</tr>
<tr>
<td>233</td>
<td>Parity bit calculation</td>
<td>1237</td>
</tr>
<tr>
<td>234</td>
<td>Audio sampling frequency versus symbol rates</td>
<td>1238</td>
</tr>
<tr>
<td>235</td>
<td>SAI interrupt sources</td>
<td>1247</td>
</tr>
<tr>
<td>236</td>
<td>SAI register map and reset values</td>
<td>1274</td>
</tr>
<tr>
<td>237</td>
<td>IPCC interface signals</td>
<td>1277</td>
</tr>
<tr>
<td>238</td>
<td>Bits used for the communication</td>
<td>1278</td>
</tr>
<tr>
<td>239</td>
<td>IPCC register map and reset values</td>
<td>1289</td>
</tr>
<tr>
<td>240</td>
<td>HSEM internal input/output signals</td>
<td>1291</td>
</tr>
<tr>
<td>241</td>
<td>HSEM register map and reset values</td>
<td>1291</td>
</tr>
<tr>
<td>242</td>
<td>Authorized AHB bus master IDs</td>
<td>1296</td>
</tr>
<tr>
<td>243</td>
<td>USB implementation</td>
<td>1302</td>
</tr>
<tr>
<td>244</td>
<td>USB implementation</td>
<td>1304</td>
</tr>
<tr>
<td>245</td>
<td>Double-buffering buffer flag definition</td>
<td>1314</td>
</tr>
<tr>
<td>246</td>
<td>Bulk double-buffering memory buffers usage</td>
<td>1314</td>
</tr>
<tr>
<td>247</td>
<td>Isochronous memory buffers usage</td>
<td>1316</td>
</tr>
<tr>
<td>248</td>
<td>Resume event detection</td>
<td>1317</td>
</tr>
<tr>
<td>249</td>
<td>Reception status encoding</td>
<td>1330</td>
</tr>
<tr>
<td>250</td>
<td>Endpoint type encoding</td>
<td>1330</td>
</tr>
<tr>
<td>251</td>
<td>Endpoint kind meaning</td>
<td>1330</td>
</tr>
<tr>
<td>252</td>
<td>Transmission status encoding</td>
<td>1331</td>
</tr>
<tr>
<td>253</td>
<td>Definition of allocated buffer memory</td>
<td>1334</td>
</tr>
<tr>
<td>254</td>
<td>USB register map and reset values</td>
<td>1335</td>
</tr>
<tr>
<td>255</td>
<td>CRS features</td>
<td>1337</td>
</tr>
<tr>
<td>256</td>
<td>Effect of low-power modes on CRS</td>
<td>1341</td>
</tr>
<tr>
<td>257</td>
<td>Interrupt control bits</td>
<td>1341</td>
</tr>
</tbody>
</table>
Table 258. CRS register map and reset values .................................................. 1346
Table 259. JTAG/Serial-wire debug port pins .................................................. 1350
Table 260. Trace port pins .............................................................................. 1351
Table 261. Single Wire Trace port pins ............................................................ 1351
Table 262. Trigger pins .................................................................................. 1351
Table 263. JTAG-DP data registers .................................................................. 1354
Table 264. Packet request ............................................................................. 1355
Table 265. ACK response ................................................................................ 1356
Table 266. Data transfer ................................................................................ 1356
Table 267. Debug port register map and reset values ..................................... 1364
Table 268. Access port register map and reset values .................................... 1372
Table 269. CPU2 CTI inputs ......................................................................... 1373
Table 270. CPU2 CTI outputs ....................................................................... 1374
Table 271. CPU1 CTI inputs ......................................................................... 1374
Table 272. CPU1 CTI outputs ....................................................................... 1374
Table 273. CTI register map and reset values .................................................. 1390
Table 274. DBGMCU register map and reset values ....................................... 1400
Table 275. CPU2 processor ROM table ........................................................... 1402
Table 276. CPU2 ROM table ......................................................................... 1402
Table 277. CPU2 processor ROM table register map and reset values .......... 1409
Table 278. CPU2 ROM table register map and reset values ......................... 1415
Table 279. CPU2 DWT register map and reset values ................................... 1427
Table 280. CPU2 BPU register map and reset values ..................................... 1436
Table 281. CPU1 ROM table ......................................................................... 1437
Table 282. CPU1 ROM table register map and reset values ......................... 1444
Table 283. CPU1 DWT register map and reset values ................................... 1456
Table 284. CPU1 ITM register map and reset values ..................................... 1465
Table 285. CPU1 FPB register map and reset values ...................................... 1472
Table 286. CPU1 ETM register map and reset values ................................... 1491
Table 287. CPU1 TPIU register map and reset values ................................... 1504
Table 288. Document revision history ............................................................ 1511
List of figures

Figure 1. System architecture .................................................. 64
Figure 2. Memory map .............................................................. 67
Figure 3. Sequential 16-bit instructions execution ......................... 81
Figure 4. Changing the Read protection (RDP) level ....................... 105
Figure 5. Radio system block diagram .................................... 132
Figure 6. CRC calculation unit block diagram ............................. 134
Figure 7. Power supply overview .......................................... 142
Figure 8. Supply configurations ............................................. 142
Figure 9. Brown-out reset waveform ...................................... 149
Figure 10. PVD thresholds ..................................................... 150
Figure 11. CPU2 boot options .................................................. 152
Figure 12. Low-power modes possible transitions ....................... 155
Figure 13. Real-time radio activity flags .................................. 173
Figure 14. Simplified diagram of the reset circuit ....................... 203
Figure 15. Clock tree ............................................................ 207
Figure 16. HSE clock sources ................................................ 208
Figure 17. LSE clock sources ................................................ 212
Figure 18. Frequency measurement with TIM16 in capture mode ...... 218
Figure 19. Frequency measurement with TIM17 in capture mode ...... 218
Figure 20. Three-volt or five-volt tolerant GPIO structure (TT or FT) 292
Figure 21. Input floating/pull up/pull down configurations .......... 297
Figure 22. Output configuration ............................................. 297
Figure 23. Alternate function configuration ................................ 298
Figure 24. High impedance-analog configuration ......................... 299
Figure 25. DMA block diagram .............................................. 330
Figure 26. DMAMUX block diagram ...................................... 353
Figure 27. Synchronization mode of the DMAMUX request line multiplexer channel ........................................... 356
Figure 28. Event generation of the DMA request line multiplexer channel ......................................................... 356
Figure 29. Interrupt block diagram ......................................... 366
Figure 30. EXTI block diagram .............................................. 375
Figure 31. Configurable event trigger logic CPU wakeup ................ 378
Figure 32. Direct event trigger logic CPU wakeup ....................... 379
Figure 33. QUADSPI block diagram ....................................... 393
Figure 34. Example of read command in quad-SPI mode .............. 394
Figure 35. Example of a DDR command in quad-SPI mode .......... 398
Figure 36. NCS when CKMODE = 0 (T = CLK period) .................. 405
Figure 37. NCS when CKMODE = 1 in SDR mode (T = CLK period) .. 405
Figure 38. NCS when CKMODE = 1 in DDR mode (T = CLK period) .. 405
Figure 39. NCS when CKMODE = 1 with an abort (T = CLK period) 406
Figure 40. ADC block diagram ................................................ 421
Figure 41. ADC clock scheme ................................................. 424
Figure 42. ADC1 connectivity ................................................ 425
Figure 43. ADC calibration .................................................... 428
Figure 44. Updating the ADC calibration factor .......................... 429
Figure 45. Mixing single-ended and differential channels ............ 430
Figure 46. Enabling / disabling the ADC .................................. 431
Figure 47. Analog to digital conversion time ............................. 436
Figure 48. Stopping ongoing regular conversions ....................... 437
Figure 49. Stopping ongoing regular and injected conversions .......................... 437
Figure 50. Injected conversion latency ......................................................... 441
Figure 51. Example of JSQR queue of context (sequence change) .................. 444
Figure 52. Example of JSQR queue of context (trigger change) ..................... 444
Figure 53. Example of JSQR queue of context with overflow before conversion 445
Figure 54. Example of JSQR queue of context with overflow during conversion 445
Figure 55. Example of JSQR queue of context with empty queue (case JQM = 0) 446
Figure 56. Example of JSQR queue of context with empty queue (case JQM = 1) 447
Figure 57. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion .......................... 447
Figure 58. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion and a new trigger occurs .................................................................................. 448
Figure 59. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs outside an ongoing conversion .......................... 448
Figure 60. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1) 449
Figure 61. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0) 449
Figure 62. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1) 450
Figure 63. Single conversions of a sequence, software trigger .......................... 452
Figure 64. Continuous conversion of a sequence, software trigger .................. 452
Figure 65. Single conversions of a sequence, hardware trigger .......................... 453
Figure 66. Continuous conversions of a sequence, hardware trigger .................. 453
Figure 67. Right alignment (offset disabled, unsigned value) ............................. 455
Figure 68. Right alignment (offset enabled, signed value) ................................. 456
Figure 69. Left alignment (offset disabled, unsigned value) ............................... 456
Figure 70. Left alignment (offset enabled, signed value) ................................. 457
Figure 71. Example of overrun (OVR) ............................................................. 458
Figure 72. AUTODLY = 1, regular conversion in continuous mode, software trigger 461
Figure 73. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) ................................................................. 461
Figure 74. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1; JDISCEN = 1) ................................................................. 462
Figure 75. AUTODLY = 1, regular continuous conversions interrupted by injected conversions 463
Figure 76. AUTODLY = 1 in auto-injected mode (JAUTO = 1) .......................... 463
Figure 77. Analog watchdog guarded area ......................................................... 464
Figure 78. ADCy_AWDx_OUT signal generation (on all regular channels) .......... 466
Figure 79. ADCy_AWDx_OUT signal generation (AWD flag not cleared by software) 467
Figure 80. ADCy_AWDx_OUT signal generation (on a single regular channel) 467
Figure 81. ADCy_AWDx_OUT signal generation (on all injected channels) .......... 467
Figure 82. 20-bit to 16-bit result truncation ..................................................... 468
Figure 83. Numerical example with 5-bit shift and rounding .............................. 468
Figure 84. Triggered regular oversampling mode (TROVS bit = 1) ..................... 470
Figure 85. Regular oversampling modes (4x ratio) ........................................... 471
Figure 86. Regular and injected oversampling modes used simultaneously .......... 472
Figure 87. Triggered regular oversampling with injection ................................. 472
Figure 88. Oversampling in auto-injected mode .............................................. 473
Figure 89. Temperature sensor channel block diagram ...................................... 474
Figure 90. VBAT channel block diagram .......................................................... 475
Figure 91. VREFINT channel block diagram .................................................... 476
Figure 92. Comparator block diagram .............................................................. 515
Figure 93. Window mode ............................................................................. 518
Figure 94. Comparator hysteresis ................................................................... 518
Figure 147. Counter timing diagram, internal clock divided by 1 .......................... 670
Figure 148. Counter timing diagram, internal clock divided by 2 .......................... 670
Figure 149. Counter timing diagram, internal clock divided by 4 .......................... 671
Figure 150. Counter timing diagram, internal clock divided by N ......................... 671
Figure 151. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .......................... 672
Figure 152. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) .......................... 672
Figure 153. Counter timing diagram, internal clock divided by 1 .......................... 674
Figure 154. Counter timing diagram, internal clock divided by 2 .......................... 674
Figure 155. Counter timing diagram, internal clock divided by 4 .......................... 675
Figure 156. Counter timing diagram, internal clock divided by N ......................... 675
Figure 157. Counter timing diagram, update event when repetition counter is not used .......................... 676
Figure 158. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 .......................... 677
Figure 159. Counter timing diagram, internal clock divided by 2 .......................... 678
Figure 160. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 .......................... 678
Figure 161. Counter timing diagram, internal clock divided by N ......................... 679
Figure 162. Counter timing diagram, update event with ARPE=1 (counter underflow) .......................... 679
Figure 163. Counter timing diagram, Update event with ARPE=1 (counter overflow) .......................... 680
Figure 164. Update rate examples depending on mode and TIMx_RCR register settings .......................... 681
Figure 165. External trigger input block .......................................................... 682
Figure 166. TIM1 ETR input circuitry ............................................................... 682
Figure 167. Control circuit in normal mode, internal clock divided by 1 .................. 683
Figure 168. TI2 external clock connection example .............................................. 684
Figure 169. Control circuit in external clock mode 1 ........................................... 685
Figure 170. External trigger input block .......................................................... 685
Figure 171. Control circuit in external clock mode 2 ........................................... 686
Figure 172. Capture/compare channel (example: channel 1 input stage) .................. 687
Figure 173. Capture/compare channel 1 main circuit ......................................... 687
Figure 174. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) ........ 688
Figure 175. Output stage of capture/compare channel (channel 4) .......................... 688
Figure 176. Output stage of capture/compare channel (channel 5, idem ch. 6) ............. 689
Figure 177. PWM input mode timing ............................................................... 691
Figure 178. Output compare mode, toggle on OC1 ............................................. 693
Figure 179. Edge-aligned PWM waveforms (ARR=8) ........................................... 694
Figure 180. Center-aligned PWM waveforms (ARR=8) ........................................... 695
Figure 181. Generation of 2 phase-shifted PWM signals with 50% duty cycle ............... 697
Figure 182. Combined PWM mode on channel 1 and 3 ....................................... 698
Figure 183. 3-phase combined PWM signals with multiple trigger pulses per period ........ 699
Figure 184. Complementary output with dead-time insertion ............................... 700
Figure 185. Dead-time waveforms with delay greater than the negative pulse .............. 700
Figure 186. Dead-time waveforms with delay greater than the positive pulse .............. 701
Figure 187. Break and Break2 circuitry overview .............................................. 703
Figure 188. Various output behavior in response to a break event on BRK (OSSI = 1) ........ 705
Figure 189. PWM output state following BRK and BRK2 pins assertion (OSSI=1) .......... 706
Figure 190. PWM output state following BRK assertion (OSSI=0) .......................... 707
Figure 191. Output redirection (BRK2 request not represented) ............................. 708
Figure 192. Clearing TIMx OCxREF ............................................................... 709
Figure 193. 6-step generation, COM example (OSSR=1) ...................................... 710
Figure 194. Example of one pulse mode ........................................................... 711
Figure 195. Retriggerable one pulse mode ......................................................... 713
Figure 196. Example of counter operation in encoder interface mode ....................... 714
Figure 197. Example of encoder interface mode with TI1FP1 polarity inverted .......... 715
Figure 198. Measuring time interval between edges on 3 signals ........................... 716
List of figures

Figure 199. Example of Hall sensor interface .................................................. 718
Figure 200. Control circuit in reset mode .......................................................... 719
Figure 201. Control circuit in Gated mode ......................................................... 720
Figure 202. Control circuit in trigger mode ....................................................... 721
Figure 203. Control circuit in external clock mode 2 + trigger mode ................. 722
Figure 204. General-purpose timer block diagram ........................................... 766
Figure 205. Counter timing diagram with prescaler division change from 1 to 2 ... 768
Figure 206. Counter timing diagram with prescaler division change from 1 to 4 ... 768
Figure 207. Counter timing diagram, internal clock divided by 1 ......................... 769
Figure 208. Counter timing diagram, internal clock divided by 2 ......................... 770
Figure 209. Counter timing diagram, internal clock divided by 4 ......................... 770
Figure 210. Counter timing diagram, internal clock divided by N ....................... 771
Figure 211. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) .... 771
Figure 212. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) .... 772
Figure 213. Counter timing diagram, internal clock divided by 1 ......................... 773
Figure 214. Counter timing diagram, internal clock divided by 2 ......................... 773
Figure 215. Counter timing diagram, internal clock divided by 4 ......................... 774
Figure 216. Counter timing diagram, internal clock divided by N ....................... 774
Figure 217. Counter timing diagram, Update event ........................................... 775
Figure 218. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 ........ 776
Figure 219. Counter timing diagram, internal clock divided by 2 ......................... 777
Figure 220. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 .. 777
Figure 221. Counter timing diagram, internal clock divided by N ....................... 778
Figure 222. Counter timing diagram, Update event with ARPE=1 (counter underflow) ... 778
Figure 223. Counter timing diagram, Update event with ARPE=1 (counter overflow) ... 779
Figure 224. Control circuit in normal mode, internal clock divided by 1 ............... 780
Figure 225. T12 external clock connection example ......................................... 780
Figure 226. Control circuit in external clock mode 1 ....................................... 781
Figure 227. External trigger input block ............................................................ 782
Figure 228. Control circuit in external clock mode 2 ....................................... 783
Figure 229. Capture/Compare channel (example: channel 1 input stage) ............. 783
Figure 230. Capture/Compare channel 1 main circuit ...................................... 784
Figure 231. Output stage of Capture/Compare channel (channel 1) ..................... 784
Figure 232. PWM input mode timing ............................................................... 786
Figure 233. Output compare mode, toggle on OC1 ......................................... 788
Figure 234. Edge-aligned PWM waveforms (ARR=8) ....................................... 789
Figure 235. Center-aligned PWM waveforms (ARR=8) ..................................... 791
Figure 236. Generation of 2 phase-shifted PWM signals with 50% duty cycle ........ 792
Figure 237. Combined PWM mode on channels 1 and 3 ................................... 793
Figure 238. Clearing TIMx OCxREF ............................................................... 794
Figure 239. Example of one-pulse mode ......................................................... 795
Figure 240. Retriggerable one-pulse mode ....................................................... 797
Figure 241. Example of counter operation in encoder interface mode .................. 798
Figure 242. Example of counter operation in encoder interface mode with TI1FP1 polarity inverted .... 799
Figure 243. Control circuit in reset mode .......................................................... 800
Figure 244. Control circuit in gated mode .......................................................... 801
Figure 245. Control circuit in trigger mode .......................................................... 802
Figure 246. Control circuit in external clock mode 2 + trigger mode ................. 803
Figure 247. Master/Slave timer example .......................................................... 804
Figure 248. Master/slave connection example with 1 channel only timers .......... 804
Figure 249. Gating TIM2 with OC1REF of TIM1 ............................................. 805
Figure 250. Gating TIM2 with Enable of TIM1 .................................................. 806
List of figures

Figure 251. Triggering TIM2 with update of TIM1 .......................................................... 807
Figure 252. Triggering TIM2 with Enable of TIM1 .......................................................... 807
Figure 253. TIM16/TIM17 block diagram ................................................................. 838
Figure 254. Counter timing diagram with prescaler division change from 1 to 2 .......... 840
Figure 255. Counter timing diagram with prescaler division change from 1 to 4 .......... 840
Figure 256. Counter timing diagram, internal clock divided by 1 .............................. 842
Figure 257. Counter timing diagram, internal clock divided by 2 .............................. 842
Figure 258. Counter timing diagram, internal clock divided by 4 .............................. 843
Figure 259. Counter timing diagram, internal clock divided by N ............................ 843
Figure 260. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) ................................................................. 844
Figure 261. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) ................................................................. 844
Figure 262. Update rate examples depending on mode and TIMx_RCR register settings ................................................................. 846
Figure 263. Control circuit in normal mode, internal clock divided by 1 .................... 847
Figure 264. TI2 external clock connection example .................................................. 847
Figure 265. Control circuit in external clock mode 1 .................................................. 848
Figure 266. Capture/compare channel (example: channel 1 input stage) ................. 849
Figure 267. Capture/compare channel 1 main circuit .............................................. 849
Figure 268. Output stage of capture/compare channel (channel 1) .......................... 850
Figure 269. Output compare mode, toggle on OC1 .................................................. 853
Figure 270. Edge-aligned PWM waveforms (ARR=8) .............................................. 854
Figure 271. Complementary output with dead-time insertion .................................. 855
Figure 272. Dead-time waveforms with delay greater than the negative pulse ........ 855
Figure 273. Dead-time waveforms with delay greater than the positive pulse .......... 856
Figure 274. Output behavior in response to a break .................................................. 858
Figure 275. Output redirection ............................................................. 860
Figure 276. 6-step generation, COM example (OSSR=1) ....................................... 861
Figure 277. Example of one pulse mode ................................................................. 862
Figure 278. Low-power timer block diagram ......................................................... 891
Figure 279. Glitch filter timing diagram ................................................................. 893
Figure 280. LPTIM output waveform, single counting mode configuration ............ 895
Figure 281. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set) ................................................................. 895
Figure 282. LPTIM output waveform, Continuous counting mode configuration .... 896
Figure 283. Waveform generation ............................................................. 897
Figure 284. Encoder mode counting sequence ....................................................... 901
Figure 285. IRTIM internal hardware connections with TIM16 and TIM17 ......... 914
Figure 286. RTC block diagram ............................................................. 917
Figure 287. Independent watchdog block diagram .............................................. 957
Figure 288. Watchdog block diagram ................................................................. 967
Figure 289. Window watchdog timing diagram ..................................................... 968
Figure 290. Block diagram ............................................................. 974
Figure 291. I2C-bus protocol ............................................................. 976
Figure 292. Setup and hold timings ................................................................. 978
Figure 293. I2C initialization flow ................................................................. 980
Figure 294. Data reception ............................................................. 981
Figure 295. Data transmission ............................................................. 982
Figure 296. Slave initialization flow ................................................................. 985
Figure 297. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0 .... 987
Figure 298. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1 .... 988
Figure 299. Transfer bus diagrams for I2C slave transmitter (mandatory events only) ................................................................. 989
Figure 300. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 0 ........................................ 990
Figure 301. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 1 ........................................ 991
Figure 302. Transfer bus diagrams for I2C slave receiver
(mandatory events only) ......................................................... 991
Figure 303. Master clock generation ........................................ 993
Figure 304. Master initialization flow ....................................... 995
Figure 305. 10-bit address read access with HEAD10R = 0 .................. 995
Figure 306. 10-bit address read access with HEAD10R = 1 .................. 996
Figure 307. Transfer sequence flow for I2C master transmitter, N ≤ 255 bytes ........................................ 997
Figure 308. Transfer sequence flow for I2C master transmitter, N > 255 bytes ........................................ 998
Figure 309. Transfer bus diagrams for I2C master transmitter
(mandatory events only) ......................................................... 999
Figure 310. Transfer sequence flow for I2C master receiver, N ≤ 255 bytes ........................................ 1001
Figure 311. Transfer sequence flow for I2C master receiver, N > 255 bytes ........................................ 1002
Figure 312. Transfer bus diagrams for I2C master receiver
(mandatory events only) ......................................................... 1003
Figure 313. Timeout intervals for t_{LOW,SEXT}, t_{LOW,MEXT} ..................... 1007
Figure 314. Transfer sequence flow for SMBus slave transmitter N bytes + PEC ........................................ 1010
Figure 315. Transfer bus diagram for SMBus slave transmitter (SBC = 1) ........................................ 1011
Figure 316. Transfer sequence flow for SMBus slave receiver N bytes + PEC ........................................ 1012
Figure 317. Bus transfer diagrams for SMBus slave receiver (SBC = 1) ........................................ 1013
Figure 318. Bus transfer diagrams for SMBus master transmitter ........................................ 1014
Figure 319. Bus transfer diagrams for SMBus master receiver ........................................ 1016
Figure 320. USART block diagram ........................................ 1039
Figure 321. Word length programming ..................................... 1042
Figure 322. Configurable stop bits .......................................... 1044
Figure 323. TC/TXE behavior when transmitting ............................ 1047
Figure 324. Start bit detection when oversampling by 16 or 8 .......... 1048
Figure 325. uart_ker_ck clock divider block diagram .................... 1051
Figure 326. Data sampling when oversampling by 16 ..................... 1052
Figure 327. Data sampling when oversampling by 8 ..................... 1053
Figure 328. Mute mode using Idle line detection ......................... 1060
Figure 329. Mute mode using address mark detection ..................... 1061
Figure 330. Break detection in LIN mode (11-bit break length - LBDL bit is set) .................. 1064
Figure 331. Break detection in LIN mode vs. Framing error detection .......................... 1065
Figure 332. USART example of synchronous master transmission .......................... 1066
Figure 333. USART data clock timing diagram in synchronous master mode
(M bits = 0) ................................................................. 1066
Figure 334. USART data clock timing diagram in synchronous master mode
(M bits = 01) ................................................................. 1067
Figure 335. USART data clock timing diagram in synchronous slave mode
(M bits = 00) ................................................................. 1068
Figure 336. ISO 7816-3 asynchronous protocol .......................... 1070
Figure 337. Parity error detection using the 1.5 stop bits .................. 1072
Figure 338. IrDA SIR ENDEC block diagram .............................. 1076
Figure 339. IrDA data modulation (3/16) - normal mode .................. 1076
Figure 340. Transmission using DMA ............................... 1078
Figure 341. Reception using DMA .......................................... 1079
Figure 342. Hardware flow control between 2 USARTs ................. 1079
Figure 343. RS232 RTS flow control ....................................... 1080
Figure 344. RS232 CTS flow control ....................................... 1081
Figure 345. Wake-up event verified (wake-up event = address match, FIFO disabled) .................. 1084
Figure 346. Wake-up event not verified (wake-up event = address match, FIFO disabled) ........................................ 1084
Figure 347. LPUART block diagram ........................................ 1126
Figure 348. LPUART word length programming ......................... 1129
Figure 349. Configurable stop bits ........................................ 1131
Figure 350. TC/TXE behavior when transmitting ......................... 1133
Figure 351. I2part_ker_ck clock divider block diagram ................. 1136
Figure 352. Mute mode using Idle line detection ....................... 1140
Figure 353. Mute mode using address mark detection ................ 1141
Figure 354. Transmission using DMA ................................... 1143
Figure 355. Reception using DMA ....................................... 1144
Figure 356. Hardware flow control between 2 LPUARTs ............... 1145
Figure 357. RS232 RTS flow control .................................. 1145
Figure 358. RS232 CTS flow control .................................. 1146
Figure 359. Wake-up event verified (wake-up event = address match, FIFO disabled) ........................................ 1149
Figure 360. Wake-up event not verified (wake-up event = address match, FIFO disabled) ........................................ 1149
Figure 361. SPI block diagram .......................................... 1178
Figure 362. Full-duplex single master/ single slave application .... 1179
Figure 363. Half-duplex single master/ single slave application .... 1180
Figure 364. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) .................................. 1181
Figure 365. Master and three independent slaves ..................... 1182
Figure 366. Multimaster application ................................... 1183
Figure 367. Hardware/software slave select management ............ 1184
Figure 368. Data clock timing diagram ................................ 1185
Figure 369. Data alignment when data length is not equal to 8-bit or 16-bit ......................................................... 1186
Figure 370. Packing data in FIFO for transmission and reception ... 1190
Figure 371. Master full-duplex communication ......................... 1193
Figure 372. Slave full-duplex communication .......................... 1194
Figure 373. Master full-duplex communication with CRC ............. 1195
Figure 374. Master full-duplex communication in packed mode .... 1196
Figure 375. NSSP pulse generation in Motorola SPI master mode ... 1199
Figure 376. TI mode transfer ............................................ 1200
Figure 377. SAI functional block diagram .............................. 1214
Figure 378. Audio frame ................................................. 1217
Figure 379. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) ........................................ 1219
Figure 380. FS role is start of frame (FSDEF = 0) ..................... 1220
Figure 381. Slot size configuration with FBOFF = 0 in SAI_xSLOTR 1221
Figure 382. First bit offset .............................................. 1221
Figure 383. Audio block clock generator overview .................... 1223
Figure 384. PDM typical connection and timing ....................... 1227
Figure 385. Detailed PDM interface block diagram .................... 1228
Figure 386. Start-up sequence .......................................... 1229
Figure 387. SAI_ADR format in TDM, 32-bit slot width ............. 1230
Figure 388. SAI_ADR format in TDM, 16-bit slot width ............. 1231
Figure 389. SAI_ADR format in TDM, 8-bit slot width ............... 1232
Figure 390. AC’97 audio frame ......................................... 1235
Figure 391. SPDIF format .............................................. 1236
Figure 392. SAI_xDR register ordering ................................ 1237
Figure 393. Data companding hardware in an audio block in the SAI ... 1241
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>394</td>
<td>Tristate strategy on SD output line on an inactive slot</td>
<td>1242</td>
</tr>
<tr>
<td>395</td>
<td>Tristate on output data line in a protocol like I2S</td>
<td>1243</td>
</tr>
<tr>
<td>396</td>
<td>Overrun detection error</td>
<td>1244</td>
</tr>
<tr>
<td>397</td>
<td>FIFO underrun event</td>
<td>1244</td>
</tr>
<tr>
<td>398</td>
<td>IPCC block diagram</td>
<td>1277</td>
</tr>
<tr>
<td>399</td>
<td>IPCC Simplex channel mode transfer timing</td>
<td>1278</td>
</tr>
<tr>
<td>400</td>
<td>IPCC Simplex - Send procedure state diagram</td>
<td>1279</td>
</tr>
<tr>
<td>401</td>
<td>IPCC Simplex - Receive procedure state diagram</td>
<td>1280</td>
</tr>
<tr>
<td>402</td>
<td>IPCC Half-duplex channel mode transfer timing</td>
<td>1281</td>
</tr>
<tr>
<td>403</td>
<td>IPCC Half-duplex - Send procedure state diagram</td>
<td>1281</td>
</tr>
<tr>
<td>404</td>
<td>IPCC Half-duplex - Receive procedure state diagram</td>
<td>1282</td>
</tr>
<tr>
<td>405</td>
<td>HSEM block diagram</td>
<td>1291</td>
</tr>
<tr>
<td>406</td>
<td>Procedure state diagram</td>
<td>1292</td>
</tr>
<tr>
<td>407</td>
<td>Interrupt state diagram</td>
<td>1295</td>
</tr>
<tr>
<td>408</td>
<td>USB peripheral block diagram</td>
<td>1305</td>
</tr>
<tr>
<td>409</td>
<td>Packet buffer areas with examples of buffer description table locations</td>
<td>1309</td>
</tr>
<tr>
<td>410</td>
<td>CRS block diagram</td>
<td>1338</td>
</tr>
<tr>
<td>411</td>
<td>CRS counter behavior</td>
<td>1339</td>
</tr>
<tr>
<td>412</td>
<td>Block diagram of debug support infrastructure</td>
<td>1350</td>
</tr>
<tr>
<td>413</td>
<td>JTAG TAP state machine</td>
<td>1353</td>
</tr>
<tr>
<td>414</td>
<td>Debug and access port connections</td>
<td>1365</td>
</tr>
<tr>
<td>415</td>
<td>Debugger connection to debug components</td>
<td>1367</td>
</tr>
<tr>
<td>416</td>
<td>Embedded cross trigger</td>
<td>1373</td>
</tr>
<tr>
<td>417</td>
<td>Mapping trigger inputs to outputs</td>
<td>1375</td>
</tr>
<tr>
<td>418</td>
<td>Cross trigger configuration example</td>
<td>1376</td>
</tr>
<tr>
<td>419</td>
<td>CPU2 CoreSight™ topology</td>
<td>1403</td>
</tr>
<tr>
<td>420</td>
<td>CPU1 CoreSight™ topology</td>
<td>1438</td>
</tr>
<tr>
<td>421</td>
<td>Trace port interface unit (TPIU)</td>
<td>1494</td>
</tr>
</tbody>
</table>
1 Documentation conventions

1.1 General information

The STM32WB55xx/STM32WB35xx devices embed an Arm® Cortex®-M4 core.

1.2 List of abbreviations for registers

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

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

---

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 STMicroelectronics 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.
- **Option bytes**: product configuration bits stored in the flash memory.
- **AHB**: advanced high-performance bus.

1.4 Availability of peripherals

For availability of peripherals and their number across all products, refer to the particular device datasheet.
2 System and memory overview

2.1 System architecture

The main system consists of 32-bit multilayer AHB bus matrix that interconnects:

Seven masters:
- CPU1 (CPU1 Cortex\textsuperscript{\textregistered}-M4 with FPU) core I-bus
- CPU1 (CPU1 Cortex\textsuperscript{\textregistered}-M4 with FPU) core D-bus
- CPU1 (CPU1 Cortex\textsuperscript{\textregistered}-M4 with FPU) core S-bus
- CPU2 (Cortex\textsuperscript{\textregistered}-M0+) core S-bus
- DMA1
- DMA2
- Radio system

Ten slaves:
- Internal Flash memory on the CPU1 (CPU1 Cortex\textsuperscript{\textregistered}-M4) ICode bus
- Internal Flash memory on CPU1 (CPU1 Cortex\textsuperscript{\textregistered}-M4) DCode bus
- Internal Flash memory on CPU2 (Cortex\textsuperscript{\textregistered}-M0+) S bus
- Internal SRAM1 (up to 192 KB)
- Internal SRAM2a (32 KB) + SRAM2b (32 KB)
- AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
- AHB2 peripherals
- Quad SPI memory interface (QUADSPI)
- AHB4 shared peripheral
- AHB5 including AHB to APB bridge and Radio peripherals (connected to APB3)

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in Figure 1.
2.1.1 S0: CPU1 (CPU1 Cortex®-M4) I-bus

This bus connects the instruction bus of the CPU1 core to the BusMatrix. This bus is used by the core to fetch instructions. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup) and the external memories through QUADSPI.

2.1.2 S1: CPU1 (CPU1 Cortex®-M4) D-bus

This bus connects the data bus of the CPU1 core to the BusMatrix. This bus is used by the core for literal load and debug access. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup) and the external memories through QUADSPI.

2.1.3 S2: CPU1 (CPU1 Cortex®-M4) S-bus

This bus connects the system bus of the CPU1 core to the BusMatrix. This bus is used by the core to access data located in a peripheral or SRAM area. The targets of this bus are SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the AHB4 peripherals and the external memories through QUADSPI.
2.1.4 S3: CPU2 (Cortex®-M0+) S-bus

This bus connects the system bus of the CPU2 core to the BusMatrix. This bus is used by the core to fetch instructions, for literal load and debug access, and access data located in a peripheral or SRAM area. The targets of this bus are the internal Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, and the AHB4 peripherals and the AHB5 peripherals including the APB3 peripherals.

2.1.5 S4, S5: DMA-bus

This bus connects the AHB master interface of the DMA to the BusMatrix. The targets of this bus are the SRAM1, SRAM2a (backup), SRAM2b (non backup), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the AHB4 peripherals and the external memories through QUADSPI.

2.1.6 S6: Radio system-bus

This bus connects the AHB master interface of the Radio system to the BusMatrix. The targets of this bus are the SRAM2a (backup) and SRAM2b (non backup).

2.1.7 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a Round Robin algorithm. The BusMatrix is composed by seven masters (CPU1: system bus, DCode bus, ICode bus, CPU2: system bus, DMA1-bus, DMA2-bus and Radio system-bus) and ten slaves (3 x Flash memory, SRAM1, SRAM2a (backup), SRAM2b (non backup), AHB1 (including APB1 and APB2), AHB2, QUADSPI, AHB4, and AHB5).

AHB/APB bridges

The two bridges AHB to APB1 and AHB to APB2 provide full synchronous connections between the AHB and the two APB buses, allowing flexible selection of the peripheral frequency.

The bridges AHB to APB3 provide an a-synchronous connections between the AHB and the APB bus, allowing flexible selection of the frequency between the AHB and peripheral.

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 SRAM1/2 and Flash memory interface). Before using a peripheral you have to enable its clock in the RCC_AHBxENR and the RCC_APBxENR registers.

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

1. It is forbidden to access QUADSPI Flash bank area before having properly configured and enabled the QUADSPI peripheral.
All the memory areas not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and register areas, refer to the following table, which gives the boundary addresses of the available peripherals.

<table>
<thead>
<tr>
<th>Bus</th>
<th>Boundary address</th>
<th>Size (bytes)</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB3</td>
<td>0xA000 1000 - 0xA000 13FF</td>
<td>1 K</td>
<td>QUADSPI</td>
<td>Section 15.5.14: QUADSPI register map on page 417</td>
</tr>
<tr>
<td></td>
<td>0xA000 0000 - 0xA000 0FFF</td>
<td>4 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x9000 0000 - 0x9FFF FFFF</td>
<td>256 M</td>
<td>QUADSPI Flash</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x6000 2000 - 0x8FFF FFFF</td>
<td>-</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x6000 1000 - 0x6000 1FFF</td>
<td>4 K</td>
<td>802.15.4 CTRL</td>
<td>-</td>
</tr>
<tr>
<td>APB3</td>
<td>0x6000 0800 - 0x6000 0FFF</td>
<td>2 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x6000 0400 - 0x6000 07FF</td>
<td>1 K</td>
<td>Radio CTRL</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x6000 0000 - 0x6000 03FF</td>
<td>1 K</td>
<td>BLE CTRL</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5800 4400 - 0x5FFFF FFF</td>
<td>128 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5800 4000 - 0x5800 43FF</td>
<td>1 K</td>
<td>FLASH</td>
<td>Section 3.10.20: FLASH register map on page 129</td>
</tr>
<tr>
<td></td>
<td>0x5800 3400 - 0x5800 3FFF</td>
<td>3 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5800 2400 - 0x5800 33FF</td>
<td>5 K</td>
<td>PKA RAM</td>
<td>Section 23.7.5: PKA register map on page 663</td>
</tr>
<tr>
<td></td>
<td>0x5800 2000 - 0x5800 23FF</td>
<td></td>
<td>PKA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5800 1C00 - 0x5800 1FFF</td>
<td>1 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x5800 1800 - 0x5800 1BFF</td>
<td>1 K</td>
<td>AES2</td>
<td>Section 22.7.18: AES register map on page 636</td>
</tr>
<tr>
<td></td>
<td>0x5800 1400 - 0x5800 17FF</td>
<td>1 K</td>
<td>HSEM</td>
<td>Section 38.4.9: HSEM register map on page 1302</td>
</tr>
<tr>
<td></td>
<td>0x5800 1000 - 0x5800 13FF</td>
<td>1 K</td>
<td>True RNG</td>
<td>Section 21.7.4: RNG register map on page 588</td>
</tr>
<tr>
<td></td>
<td>0x5800 0C00 - 0x5800 0FFF</td>
<td>1 K</td>
<td>IPCC</td>
<td>Section 37.4.9: IPCC register map on page 1289</td>
</tr>
<tr>
<td></td>
<td>0x5800 0800 - 0x5800 0BFF</td>
<td>1 K</td>
<td>EXTI</td>
<td>Section 14.6.17: EXTI register map on page 391</td>
</tr>
<tr>
<td></td>
<td>0x5800 0400 - 0x5800 07FF</td>
<td>1 K</td>
<td>PWR</td>
<td>Section 6.6.24: PWR register map and reset value table on page 194</td>
</tr>
<tr>
<td></td>
<td>0x5800 0000 - 0x5800 03FF</td>
<td>1 K</td>
<td>RCC</td>
<td>Section 8.4.49: RCC register map on page 285</td>
</tr>
<tr>
<td>Bus</td>
<td>Boundary address</td>
<td>Size (bytes)</td>
<td>Peripheral</td>
<td>Peripheral register map</td>
</tr>
<tr>
<td>------</td>
<td>------------------------</td>
<td>--------------</td>
<td>------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>0x5006 0400 - 0x57FF FFFF</td>
<td>-</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x5006 0000 - 0x5006 03FF</td>
<td>1 K</td>
<td>AES1</td>
<td>Section 22.7.18: AES register map on page 636</td>
</tr>
<tr>
<td></td>
<td>0x5004 0000 - 0x5004 03FF</td>
<td>1 K</td>
<td>ADC</td>
<td>Section 16.9: ADC register map on page 507</td>
</tr>
<tr>
<td></td>
<td>0x4800 1C00 - 0x4800 1FFF</td>
<td>1 K</td>
<td>GPIOH</td>
<td>Section 9.5.12: GPIO register map on page 308</td>
</tr>
<tr>
<td>AHB2</td>
<td>0x4800 1400 - 0x4800 1BFF</td>
<td>3 K</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4800 1000 - 0x4800 13FF</td>
<td>1 K</td>
<td>GPIOE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4800 0C00 - 0x4800 0FFF</td>
<td>1 K</td>
<td>GPIOD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4800 0800 - 0x4800 0BFF</td>
<td>1 K</td>
<td>GPIOC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4800 0400 - 0x4800 07FF</td>
<td>1 K</td>
<td>GPIOB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4800 0000 - 0x4800 03FF</td>
<td>1 K</td>
<td>GPIOA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4800 0000 - 0x4800 15FF</td>
<td>127 M</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4002 4000 - 0x4002 43FF</td>
<td>1 K</td>
<td>TSC</td>
<td>Section 20.6.11: TSC register map on page 575</td>
</tr>
<tr>
<td></td>
<td>0x4002 3400 - 0x4002 3FFF</td>
<td>3 K</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4002 3000 - 0x4002 33FF</td>
<td>1 K</td>
<td>CRC</td>
<td>Section 5.4.6: CRC register map on page 139</td>
</tr>
<tr>
<td></td>
<td>0x4002 0C00 - 0x4002 2FFF</td>
<td>9 K</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4002 0800 - 0x4002 0BFF</td>
<td>1 K</td>
<td>DMAMUX</td>
<td>Section 12.6.7: DMAMUX register map on page 363</td>
</tr>
<tr>
<td></td>
<td>0x4002 0400 - 0x4002 07FF</td>
<td>1 K</td>
<td>DMA2</td>
<td>Section 11.6.7: DMA register map on page 346</td>
</tr>
<tr>
<td></td>
<td>0x4002 0000 - 0x4002 03FF</td>
<td>1 K</td>
<td>DMA1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4001 5800 - 0x4001 FFFF</td>
<td>42 K</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
### Table 1. Memory map and peripheral register boundary addresses (continued)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Boundary address</th>
<th>Size (bytes)</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB2</td>
<td>0x4001 5400 - 0x4001 57FF</td>
<td>1 K</td>
<td>SAI1</td>
<td>Section 36.6.19: SAI register map on page 1274</td>
</tr>
<tr>
<td></td>
<td>0x4001 4C00 - 0x4001 53FF</td>
<td>2 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 4800 - 0x4001 4BFF</td>
<td>1 K</td>
<td>TIM17</td>
<td>Section 26.4.23: TIM16/TIM17 register map on page 887</td>
</tr>
<tr>
<td></td>
<td>0x4001 4400 - 0x4001 47FF</td>
<td>1 K</td>
<td>TIM16</td>
<td>Section 26.4.23: TIM16/TIM17 register map on page 887</td>
</tr>
<tr>
<td></td>
<td>0x4001 3C00 - 0x4001 43FF</td>
<td>2 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 3800 - 0x4001 3BFF</td>
<td>1 K</td>
<td>USART1</td>
<td>Section 33.8.15: USART register map on page 1121</td>
</tr>
<tr>
<td></td>
<td>0x4001 3400 - 0x4001 37FF</td>
<td>1 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 3000 - 0x4001 33FF</td>
<td>1 K</td>
<td>SPI1</td>
<td>Section 35.6.8: SPI register map on page 1211</td>
</tr>
<tr>
<td></td>
<td>0x4001 2C00 - 0x4001 2FF</td>
<td>1 K</td>
<td>TIM1</td>
<td>Section 24.4.30: TIM1 register map on page 762</td>
</tr>
<tr>
<td></td>
<td>0x4001 0400 - 0x4001 2BFF</td>
<td>10 K</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>0x4001 0200 - 0x4001 03FF</td>
<td></td>
<td>COMP</td>
<td>Section 18.6.3: COMP register map on page 526</td>
</tr>
<tr>
<td></td>
<td>0x4001 0100 - 0x4001 01FF</td>
<td>1 K</td>
<td>SYSCFG</td>
<td>Section 10.2.17: SYSCFG register map on page 325</td>
</tr>
<tr>
<td></td>
<td>0x4001 0030 - 0x4001 00FF</td>
<td></td>
<td>VREFBUF</td>
<td>Section 17.5.3: VREFBUF register map on page 512</td>
</tr>
<tr>
<td></td>
<td>0x4001 0000 - 0x4001 002F</td>
<td></td>
<td>SYSCFG</td>
<td>Section 10.2.17: SYSCFG register map on page 325</td>
</tr>
</tbody>
</table>
### Table 1. Memory map and peripheral register boundary addresses (continued)

<table>
<thead>
<tr>
<th>Bus</th>
<th>Boundary address</th>
<th>Size (bytes)</th>
<th>Peripheral</th>
<th>Peripheral register map</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB1</td>
<td>0x4000 9800 - 0x4000 FFFF</td>
<td>26 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 9400 - 0x04000 97FF</td>
<td>1 K LPTIM2</td>
<td>Section 27.7.11: LPTIM register map on page 913</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 8400 - 0x04000 93FF</td>
<td>4 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 8000 - 0x04000 83FF</td>
<td>1 K LPUART1</td>
<td>Section 34.7.13: LPUART register map on page 1175</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 7C00 - 0x04000 7FFF</td>
<td>1 K LPTIM1</td>
<td>Section 27.7.11: LPTIM register map on page 913</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 7000 - 0x04000 7BFF</td>
<td>3 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 6C00 - 0x04000 6FFF</td>
<td>1 K USB SRAM</td>
<td>Section 39.6.3: USB register map on page 1335</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 6800 - 0x04000 6BFF</td>
<td>1 K USB FS</td>
<td>Section 39.6.3: USB register map on page 1335</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 6400 - 0x04000 67FF</td>
<td>1 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 6000 - 0x04000 63FF</td>
<td>1 K CRS</td>
<td>Section 40.7.5: CRS register map on page 1346</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 5C00 - 0x04000 5FFF</td>
<td>1 K I2C3</td>
<td>Section 32.9.12: I2C register map on page 1034</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 5800 - 0x04000 5BFF</td>
<td>1 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 5400 - 0x04000 57FF</td>
<td>1 K I2C1</td>
<td>Section 32.9.12: I2C register map on page 1034</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 3C00 - 0x04000 53FF</td>
<td>6 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 3800 - 0x04000 3BFF</td>
<td>1 K SPI2</td>
<td>Section 35.6.8: SPI register map on page 1211</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 3400 - 0x04000 37FF</td>
<td>1 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 3000 - 0x04000 33FF</td>
<td>1 K IWDG</td>
<td>Section 30.4.6: IWDG register map on page 965</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 2C00 - 0x04000 2FFF</td>
<td>1 K WWGD</td>
<td>Section 31.5.4: WWGD register map on page 971</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 2800 - 0x04000 2BFF</td>
<td>1 K RTC &amp; TAMP</td>
<td>Section 29.7.21: RTC register map on page 955</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 2400 - 0x04000 27FF</td>
<td>1 K LCD</td>
<td>Section 19.6.8: LCD register map on page 555</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 0400 - 0x04000 23FF</td>
<td>8 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4000 0000 - 0x4000 03FF</td>
<td>1 K TIM2</td>
<td>Section 25.4.25: TIMx register map on page 834</td>
<td></td>
</tr>
<tr>
<td>AHB4</td>
<td>0x2003 8000 - 0x2003 FFFF</td>
<td>32 K SRAM2b</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2003 0000 - 0x2003 7FFF</td>
<td>32 K SRAM2a</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2000 3000 - 0x2002 FFFF</td>
<td>36 K Reserved</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>AHB1</td>
<td>0x2000 0000 - 0x2002 FFFF</td>
<td>192(1) K SRAM1</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
2.2.3 Bit banding

The CPU1 map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

The AHB1, APB1, APB2 peripheral registers and the SRAM1, SRAM2a and SRAM2b are mapped to a bit-band region, hence single bit-band write and read operations are allowed. The operations are only available for CPU1 accesses, and not form other bus masters (e.g. DMA).

The peripheral bit-band alias is located from address 0x4200 0000 to 0x42FF FFFF.

The SRAM bit-band alias is located from address 0x2200 0000 to 0x227F FFFF.

A mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region. The mapping formula is:

\[
\text{bit\_word\_addr} = \text{bit\_band\_base} + (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4),
\]

where:

- \(\text{bit\_word\_addr}\) is the address of the word in the alias memory region that maps to the targeted bit.
- \(\text{bit\_band\_base}\) is the starting address of the alias region.
- \(\text{byte\_offset}\) is the number of the byte in the bit\_band region that contains the targeted bit.
- \(\text{bit\_number}\) is the bit position (0-7) of the targeted bit.

Example

The following example shows how to map bit [2] of the byte located at SRAM1 address 0x2000 0300 to the alias region.

\[
0x2200\ 6008 = 0x2200\ 0000 + 0x0300 \times 32 + 2 \times 4
\]

Writing to address 0x2200 6008 has the same effect as a read-modify-write operation on bit [2] of the byte at SRAM1 address 0x2000 0300.
Reading address 0x2200 6008 returns the value 0x01 or 0x00 of bit [2] of the byte at SRAM1 address 0x2000 0300.

For more information on bit-band, refer to the Cortex®-M4 programming manual.

## 2.3 Boot configuration

Three different CPU1 boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the User options, as shown in Table 2.

### Table 2. Boot modes

<table>
<thead>
<tr>
<th>nBOOT1 FLASH_OPTR[23]</th>
<th>nBOOT0 FLASH_OPTR[27]</th>
<th>BOOT0 pin PH3</th>
<th>nSWBOOT0 FLASH_OPTR[26]</th>
<th>Main flash empty(1)</th>
<th>Boot memory space alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Main flash memory is selected as boot area</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>System memory is selected as boot area</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>Main flash memory is selected as boot area</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>Embedded SRAM1 is selected as boot area</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>System memory is selected as boot area</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>System memory is selected as boot area</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>System memory is selected as boot area</td>
</tr>
</tbody>
</table>

1. A flash empty check mechanism is implemented to force the boot from system flash if the first flash memory location is not programmed (0xFFFF FFFF) and if the boot selection was configured to boot from the main flash.

The values on both BOOT0 and BOOT1 are latched after a reset. It is up to the user to provide the correct value for the required boot mode.

The BOOT0 and BOOT1 are also re-sampled when exiting Standby mode. Consequently they must be kept in the required boot mode. After the startup delay, the CPU1 fetches the top-of-stack from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004.

Depending on the selected boot mode, main flash, system flash, or SRAM1 memories are accessible as follows:

- **Boot from main flash memory:** the main flash memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is accessible even from its physical address 0x0800 0000. In other words, the flash memory content can be accessed starting from address 0x0000 0000 or 0x0800 0000.
- **Boot from system flash memory:** the system flash memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is also still accessible from its physical address 0x1FFF 0000.
- **Boot from SRAM:** the memory is aliased in the CPU1 boot memory space at address 0x0000 0000, and is accessible even from its physical address 0x2000 0000.
Empty check

An internal empty check flag (the EMPTY bit of the FLASH access control register (FLASH_ACR)) is available for easy programming of virgin devices by the boot loader. This flag is used when BOOT0 pin is defining main flash as the target boot area. When the flag is set, the device is considered as empty, and the system memory (boot loader) is selected instead of the main flash as a boot area, to allow user to program the memory. Therefore, some of the GPIOs are reconfigured from the high-Z state. Refer to AN2606 for more details concerning the bootloader and GPIO configuration in system memory boot mode. It is possible to disable this feature by configuring the option bytes to force boot from the main flash memory (nSWBOOT0 = 0, nBOOT0 = 1).

This empty check flag is updated only during the loading of option bytes: it is set when the content of the address 0x0800 0000 is read as 0xFFFF FFFF, otherwise it is cleared. A power reset or setting the 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 be written directly by software.

CPU1 physical remap

Following CPU1 boot the application software can modify the memory map at address 0x0000 0000. This modification is performed by programming the SYSCFG memory remap register (SYSCFG_MEMRMP) in the SYSCFG controller.

The following memories can be remapped:
- Main flash memory
- System flash memory
- SRAM
- Quad SPI memory

Embedded boot loader

The embedded boot loader is located in the system flash memory, programmed by ST during production. It is used to program the flash memory using one of the following device interfaces:
- USART1 on pins PA9 and PA10
- I2C1 on pins PB6 and PB7.
- I2C3 on pins PC0 and PC1
- SPI1 on pins PA4, PA5, PA6 and PA7
- SPI2 on pins PB12, PB13, PB14 and PB15
- USB FS on pins PA11 and PA12.

2.4 CPU2 boot

Following a device reset the CPU2 will only boot after CPU1 has set the C2BOOT bit in the PWR control register 4 (PWR_CR4). The C2BOOT value is retained in Standby mode and the CPU2 will boot accordingly when exit from Standby.

The CPU2 will boot from it boot reset vector as defined by the flash user option C2OPT and SBRV.

The CPU2 may boot from anywhere in user flash or SRAM1/SRAM2a/SRAM2b.
**CPU2 safe boot**

When, after a reset, the User options are not valid and the BOOT0 and BOOT1 select CPU1 to boot from main flash memory, the CPU2 boots from a safe boot vector in main flash memory at address 0x080F F000.

The safe boot can be used to restore the last known user options from an copied image.

### 2.5 CPU2 SRAM fetch disable

CPU2 execution from SRAM can be disabled by the C2RFD bit in SYSCFG register. Disabling CPU2 execution from SRAM improves robustness of the CPU2 software.
3 Embedded flash memory (FLASH)

3.1 Introduction

The flash memory interface manages CPU1 (CPU1 Cortex®-M4) AHB ICode and DCode accesses and the CPU2 (Cortex® -M0+) AHB to the flash memory. It implements the access arbitration between the two CPUs, the erase and program flash memory operations, the read and write protection, and the 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 1 Mbyte of flash memory single bank architecture
- Memory organization: 1 bank
  - main memory: up to 1 Mbyte
  - page size: 4 Kbytes
- 72-bit wide data read (64 bits plus 8 ECC bits)
- 72-bit wide data write (64 bits plus 8 ECC bits)
- Page erase (4 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 area selected by option (PCROP)
- CPU2 Security area
- Flash empty check
- Program and Erase suspension feature
- Prefetch on CPU1 ICODE and CPU2 S-bus
- CPU1 instruction cache: 32 cache lines of 4 x 64 bits on ICode (1 Kbyte RAM)
- CPU1 data cache: eight cache lines of 4 x 64 bits on DCode (256 bytes RAM)
- CPU2 instruction cache: four cache lines of 1 x 64 bits on S-bus (32 bytes RAM)
- Error code correction (ECC): eight bits for 64-bit
- Option byte loader
3.3 FLASH functional description

3.3.1 Flash memory organization

The 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. It includes:

A main memory block containing up to 256 pages for STM32WB55xx, up to 128 pages for STM32WB35xx of 4 Kbytes, each page with eight rows of 512 bytes.

An information block containing:

- System memory from which the CPU1 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, USB, I2C1, I2C3, SPI1, SPI2 (available only on STM32WB55xx). It is programmed by STMicroelectronics when the device is manufactured, 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.

- Option bytes for user configuration.

The memory is based on a main area and an information block as shown in Table 3.

<table>
<thead>
<tr>
<th>Area</th>
<th>Addresses</th>
<th>Size (bytes)</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main memory(1)</td>
<td>0x0800 0000 - 0x0800 0FFF</td>
<td>4 K</td>
<td>Page 0</td>
</tr>
<tr>
<td></td>
<td>0x0800 1000 - 0x0800 1FFF</td>
<td>4 K</td>
<td>Page 1</td>
</tr>
<tr>
<td></td>
<td>0x0800 2000 - 0x0800 2FFF</td>
<td>4 K</td>
<td>Page 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0807 E000 - 0x0807 EFFF</td>
<td>4 K</td>
<td>Page 126</td>
</tr>
<tr>
<td></td>
<td>0x0807 F000 - 0x0807 FFFF</td>
<td>4 K</td>
<td>Page 127</td>
</tr>
<tr>
<td></td>
<td>0x0808 0000 - 0x0808 0FFF</td>
<td>4 K</td>
<td>Page 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x080F E000 - 0x080F EFFF</td>
<td>4 K</td>
<td>Page 254</td>
</tr>
<tr>
<td></td>
<td>0x080F F000 - 0x080F FFFF</td>
<td>4 K</td>
<td>Page 255</td>
</tr>
<tr>
<td>Information block</td>
<td>0x1FFF 0000 - 0x1FFF 6FFF</td>
<td>28 K</td>
<td>System memory</td>
</tr>
<tr>
<td></td>
<td>0x1FFF 7000 - 0x1FFF 73FF</td>
<td>1 K</td>
<td>OTP area</td>
</tr>
<tr>
<td></td>
<td>0x1FFF 8000 - 0x1FFF 807F</td>
<td>128</td>
<td>Option bytes</td>
</tr>
</tbody>
</table>

1. Pages 128 to 255 available on STM32WB55xx only.
3.3.2 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 flash main memory area when, for instance, no user code has been programmed.

The flash main memory empty check status can be read from the EMPTY bit in Flash memory access control register (FLASH_ACR). Software can modify the flash main memory empty status by writing to the EMPTY bit.

3.3.3 Error code correction (ECC)

Data in flash memory words are 72-bit 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 memory 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 memory 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] in 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 (0xFFF 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 Read access latency

To correctly read data from flash memory, the number of wait states (LATENCY) must be correctly programmed in the Flash memory access control register (FLASH_ACR) according to the frequency of the flash memory (HCLK4) clock and the internal voltage range of the device VCORE. Refer to Section 6.1.6: Dynamic voltage scaling management.
Table 4 shows the correspondence between wait states and flash clock frequency.

Table 4. Number of wait states vs, Flash memory clock (HCLK4) frequency

<table>
<thead>
<tr>
<th>Wait states (WS) (LATENCY)</th>
<th>HCLK4 (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V_{CORE} Range 1</td>
</tr>
<tr>
<td>0 WS (1 HCLK cycles)</td>
<td>≤ 18</td>
</tr>
<tr>
<td>1 WS (2 HCLK cycles)</td>
<td>≤ 36</td>
</tr>
<tr>
<td>2 WS (3 HCLK cycles)</td>
<td>≤ 54</td>
</tr>
<tr>
<td>3 WS (4 HCLK cycles)</td>
<td>≤ 64</td>
</tr>
</tbody>
</table>

After POR on reset, the HCLK4 clock frequency is 4 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH_ACR register.

When woken-up from Standby, the HCLK4 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 software sequences described below 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 in the Flash memory 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 from the Flash memory 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 in the RCC_CFGR register.
4. If needed, modify the CPU clock prescaler by writing the SHDHPRE bits in RCC_EXTCFGR.
5. Optionally, check that the new System clock source or/and the new flash memory clock prescaler value is/are taken into account by reading the clock source status (SWS bits) in the RCC_CFGR register, or/and the AHB prescaler value (SHDHPREF bit), in the RCC_EXTCFGR register.

Decreasing the CPU frequency

1. Modify the System clock source by writing the SW bits in the RCC_CFGR register.
2. If needed, modify the flash memory clock prescaler by writing the SHDHPRE bits in RCC_EXTCFGR.
3. Check that the new System clock source or/and the new flash memory clock prescaler value is/are taken into account by reading the clock source status (SWS bits) in the RCC_CFGR register, or/and the AHB prescaler value (SHDHPREF bit), in the
RCC_EXTCFGGR register, and wait until the programmed new system clock source or new flash memory clock prescaler value is/are read.

4. Program the new number of wait states to the LATENCY bits in Flash memory 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 from the Flash memory access control register (FLASH_ACR).

### 3.3.5 Adaptive real-time memory accelerator (ART Accelerator)

The proprietary Adaptive real-time (ART) memory accelerator is optimized for STM32 industry-standard Arm® Cortex®-M4 with FPU processors. It balances the inherent performance advantage of the Arm® Cortex®-M4 with FPU over flash memory technologies, which normally require the processor to wait for the flash memory at higher operating frequencies.

To release the processor full performance, the accelerator implements an instruction prefetch queue and branch cache that increases program execution speed from the 64-bit flash memory. Based on CoreMark® benchmark, the performance achieved thanks to the ART Accelerator™ is equivalent to 0 wait state program execution from flash memory at a CPU frequency up to 64 MHz.

#### Instruction prefetch

The CPU1 fetches the instruction over the ICode bus and the literal pool (constant/data) over the DCode bus. The prefetch block aims at increasing the efficiency of ICode bus accesses.

The CPU2 fetches the instruction and the literal pool (constant/data) over the S-bus. The prefetch block aims at increasing the efficiency of S-bus accesses.

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 CPU1 ICode bus or CPU2 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 in the Flash memory access control register (FLASH_ACR) for the CPU1 or Flash memory CPU2 access control register (FLASH_C2ACR) for the CPU2. This feature is useful if at least one wait state is needed to access the flash memory.

*Figure 3* shows the execution of sequential 16-bit instructions with and without prefetch when three WS are 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.

**CPU1 Instruction cache memory (I-Cache)**

To limit the CPU1 time lost due to jumps, it is possible to retain 32 lines of 4*64 bits (1 Kbyte) in an instruction cache memory. This feature can be enabled for the CPU1 by setting the instruction cache enable (ICEN) bit in the Flash memory 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 have been 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 enable after system reset.

**CPU1 Data cache memory (D-Cache)**

CPU1 literal pools are fetched from flash memory through the DCode bus during the execution stage of the CPU pipeline. Each CPU1 DCode bus read access fetches 64 bits that are saved in a current buffer. The CPU pipeline is consequently stalled until the requested literal pool is provided. To limit the time lost due to literal pools, accesses through the AHB databus DCode have priority over accesses through the AHB instruction bus ICode.

If some literal pools are frequently used, the CPU1 data cache memory can be enabled by setting the data cache enable (DCEN) bit in the Flash memory access control register (FLASH_ACR). This feature works like the instruction cache memory, but the retained data size is limited to 8 lines of 4*64 bits (256 bytes).

The Data cache memory is enabled after system reset.

*Note:* The D-Cache is active only when data is requested by the CPU (not by DMAs).

Data in option bytes block are not cacheable.

**CPU2 cache memory (S-bus)**

To limit the CPU2 time lost due to jumps, it is possible to retain four lines of 1*64 bits (32 bytes) in an instruction cache memory. This feature can be enabled for the CPU2 by setting the instruction cache enable (ICEN) bit in the Flash memory CPU2 access control register (FLASH_C2ACR). 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 have been 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 enable after system reset.

CPU2 literal pools are fetched from flash memory through the S-bus during the execution stage of the CPU pipeline. Each CPU2 S-bus read access fetches 64 bits that are 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 memory program and erase operations

The embedded flash memory can be programmed using in-circuit or in-application programming.

The in-circuit programming (ICP) method is used to update the entire contents of the flash memory, using the JTAG, SWD protocol or the supported interfaces by the System boot loader, to load the user application for both the CPU1 and CPU2, 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 (e.g. I/Os, USB, UART, I2C, SPI) 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 stalls the bus. The read operation proceeds correctly once the program/erase operation is completed.

Note: In a multi-CPU system it is good practice to use semaphores to manage flash memory program and erase operations, and prevent simultaneous operations by the CPUs.

Secure system flash memory programming

The secure CPU2 application can only be programmed by in-application programming (IAP) running on the secure CPU2. Only the secure CPU2 is able to download programming data into secure part of the memory. Secure 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 in-circuit programming (ICP) System boot loader is able to communicate with the secure CPU2 IAP, to download programming data into secure memory.

Unlocking the flash memory

After reset, write is not allowed in the Flash memory control register (FLASH_CR) or Flash memory CPU2 control register (FLASH_C2CR) to protect the flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence is used to unlock these registers:

1. Write KEY1 = 0x4567 0123 in the Flash memory key register (FLASH_KEYR)
2. Write KEY2 = 0xCDEF 89AB in the Flash memory key register (FLASH_KEYR).

Any wrong sequence locks up 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 BSY bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR) is set. Any
attempt to write to these registers with the BSY bit set causes the AHB bus to stall until this bit is cleared.

**Caution:** Due to the structure of the memory (64 bits of data associated to 8 bits of ECC), interrupted operations can result in corrupted data and/or inconsistent ECC bits. The concerned address can trigger an ECC interrupt or an NMI when accessed later on. Consequently, handle write/erase operations with care, and ensure that ECC events are managed by firmware.

### 3.3.7 Flash main memory erase sequences

The 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, OTP and option bytes).

**Flash memory page erase**

The CPU1 is able to page erase only the non-secure part of the user flash. The secure CPU2 can page erase both the secure and non-secure parts of the user flash memory.

A page erase starts only when allowed by the PESD bit in the *Flash memory status register (FLASH_SR)* and in the *Flash memory CPU2 status register (FLASH_C2SR)*.

When a page is protected by PCROP or WRP it is not erased.

**Table 5. Page erase overview**

<table>
<thead>
<tr>
<th>Page</th>
<th>PCROP</th>
<th>WRP</th>
<th>PCROP_RDP</th>
<th>Comment</th>
<th>WRPERR</th>
<th>CPU1 bus error</th>
<th>CPU2 bus error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non secure</td>
<td>No</td>
<td>No</td>
<td></td>
<td>Page is erased</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
<td>Page erase aborted (no page erase started)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Requested by CPU2, secure page is erased</td>
<td>No</td>
<td>N/A</td>
<td>No</td>
</tr>
<tr>
<td>Secure</td>
<td>No</td>
<td>No</td>
<td>x</td>
<td>Requested by CPU1, secure page erase is aborted (no secure page erase started)</td>
<td>No</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>No</td>
<td></td>
<td>Page erase aborted (no page erase started)</td>
<td>Yes(1)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. WRPERR is generated only when PER is requested by CPU2 (Cortex-M0+). There is no WRPERR when PER is requested by CPU1.

To erase a page (4 Kbytes), follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*.
2. Check that flash memory program and erase operations are allowed by checking the PESD bit in the *Flash memory status register (FLASH_SR)* or *Flash memory CPU2 status register (FLASH_C2SR)*.

---

84/1530 RM0434 Rev 13
status register (FLASH_C2SR) (these checks are recommended even if status can change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting page erase).

2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.

3. Check that bit CFGBSY in the FLASH_xxSR register is cleared.

4. Set the PER bit and select the page to erase (PNB) in the Flash memory control register (FLASH_CR) or Flash memory CPU2 control register (FLASH_C2CR).

5. Set the STRT bit in the FLASH_xxCR register.

6. Wait until bit CFGBSY 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

A flash memory mass erase by the CPU1 is ignored and a bus error is generated.

When PCROP or WRP is enabled, any flash memory mass erase is aborted and no erase started.

Table 6. Mass erase overview

<table>
<thead>
<tr>
<th>PCROP</th>
<th>WRP</th>
<th>PCROP_RDP</th>
<th>Comment</th>
<th>WRPERRO</th>
<th>CPU1 bus error</th>
<th>CPU2 bus error</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>x</td>
<td>Requested by secure CPU2, flash memory is mass erased</td>
<td>No</td>
<td>N/A</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td></td>
<td>Requested by secure CPU2, mass erase aborted (no erase started)</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td></td>
<td>Requested by secure CPU2, mass erase aborted (no erase started)</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Requested by CPU1, mass erase aborted (no erase started)</td>
<td>No</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>Requested by CPU1, mass erase aborted (no erase started)</td>
<td>No</td>
<td>Yes</td>
<td>N/A</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 BSY bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR).

2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.

3. Check that bit CFGBSY in the FLASH_xxSR register is cleared.

4. Set the MER bit in the Flash memory control register (FLASH_CR) or Flash memory CPU2 control register (FLASH_C2CR).

5. Set the STRT bit in the FLASH_xxCR register.

6. Wait until bit CFGBSY 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 (a double word, 64 bits plus ECC, 8 bits) at a time. Programming in a previously programmed double word is only allowed when programming an all 0 value. It is not allowed to program any other value in a previously programmed double word, any attempt sets PROGERR flag in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR), except when programming an already programmed double word with an all 0 value.

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) sets SIZERR flag in the FLASH_xxSR register.
- Any attempt to write a double word that is not aligned with a double word address sets PGAERR flag in the FLASH_xxSR register.

Only the secure CPU2 is able to download programming data into the secure part of the memory. A flash memory programming by the CPU1 in the secure flash memory area is ignored and a bus error is generated.

Standard programming

The flash memory programming sequence in standard mode is as follows:

1. Check that no flash main memory operation is ongoing by checking the BSY bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR).
   Check that flash memory program and erase operations are allowed by checking the PESD bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR) (these checks are recommended even if status may change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting programming).

2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.

3. Check that bit CFGBSY in the FLASH_xxSR register is cleared.

4. Set the PG bit in the Flash memory control register (FLASH_CR) or Flash memory CPU2 control register (FLASH_C2CR).

5. 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.

6. Wait until bit CFGBSY is cleared.

7. Check that EOP flag is set in the FLASH_xxSR register (meaning that the programming operation has succeeded), and clear it by software.

8. Clear the PG bit in the FLASH_xxSR register if there are no more programming requests anymore.
Note: When the flash memory interface has received a good sequence (a double word), programming is automatically launched and BSY 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

This mode allows to program a row or 64 double words (512 bytes). It reduces the page programming time by eliminating the need for verifying the flash memory locations before they are programmed, and avoids rising and falling times of high voltage for each double word. During fast programming, the clock frequency (HCLK4) must be at least 8 MHz.

Only the main memory can be programmed in fast programming mode.

Fast row programming must be performed by executing software from SRAM and disabling interrupts when not relocating the CPU interrupt vector table. A read access from the CPU requesting row programming causes a bus error. A read from any other source (the other CPU or DMA) is stalled until the row programming finishes (standard double word programming does not cause a bus error to the requesting CPU, but stalls any read until standard programming finishes).

The flash main memory programming sequence in standard mode is described below:

1. Perform a mass erase. If not, PGSERR is set.

2. Check that no flash main memory operation is ongoing by checking the BSY bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR).

   Check that flash memory program and erase operation is allowed by checking the PESD bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR) (these checks are recommended even if status may change due to flash memory operation requests by the other CPU, to limit the risk of receiving a bus error when starting programming).

3. Check and clear all error programming flags due to previous programming.

4. Check that bit CFGBSY in the FLASH_xxSR register is cleared.

5. Set the FSTPG bit in Flash memory control register (FLASH_CR) or Flash memory CPU2 control register (FLASH_C2CR).

6. Write the 64 double words to program a row (512 bytes).

7. Wait until bit CFGBSY is cleared.

8. Check that EOP flag is set in the FLASH_xxSR register (meaning that the programming operation has succeed), and clear it by software.

9. Clear the FSTPG bit in the FLASH_xxSR register 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 BSY 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 64 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 64 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 signaled by flags

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 with values different from all 0 has not been previously erased, except if the value to program is all 0. If any other error (SIZERR, PAGERR, PGSERR, WRPERR) occurs the PROGRERR may not be set even if there is a word re-programming without erase error.

- **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:
  - Alignment programming errors are individually checked per CPU. No checks are available for simultaneous multi-CPU programming. HSEM or other software mechanisms shall be used to prevent simultaneous multi CPU programming.
  - In standard programming: the first word to be programmed is not aligned with a double word address, or the second word does not belong to the same double word address.
  - In fast programming: the data to program does not 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: data is written when PG and FSTPG are cleared.
  - In the standard programming sequence or the fast programming sequence: MER 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 MER is set.
In the page erase sequence: PG, FSTPG, and MER 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.

In the fast programming sequence: if the row to be programmed is on a page different from the one previously erased. If a mass erase has been done it is allowed to program rows on higher order pages, but not on lower order.

**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 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, except when requested by the secure CPU2.

**MISSERR**: Fast programming data miss error

In fast programming: all the data must be written successively. MISSERR is set if the previous data programming 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 7 µs, 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 is set in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR):

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

If this case, if the error interrupt enable bit ERRIE is set in the Flash memory control register (FLASH_CR) or in the Flash memory CPU2 control register (FLASH_C2CR), an interrupt is generated and the operation error flag OPERR is set in the FLASH_xxSR register.

*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 errors causing a bus error

Some error conditions, listed below, do not generate an error flag but a bus error instead.

- AHB write to any page when RDP level 1 and boot is performed from system flash memory or SRAM1
- AHB write when flash memory is powered down
- Read or write from flash memory through the debugger
- Reading from flash memory when fast row programming is ongoing, from the source which requested the fast row programming.
- Requesting a new programming request, when the previous one has not finished.
- FLASH_CR register write between the two accesses of a double word programming.
- FLASH_CR register write when PESDx is active (set).
- Writing a wrong key in FLASH_KEYR or FLASH_OPTKEYR register
- Any subsequent write to FLASH_KEYR or FLASH_OPTKEYR after unlocking the respective feature

PGSERR and PGAERR in a page-based row programming

When performing a fast programming Table 7 describes how PGSERR and PGAERR are handled.

<table>
<thead>
<tr>
<th>Last page / row</th>
<th>Current page / row</th>
<th>MER active</th>
<th>PER active</th>
</tr>
</thead>
</table>

When after a system reset neither MER nor PER is performed, any programming attempt causes PGAERR and PGSERR errors.

Programming and caches

If a flash memory write access impacts data in the data cache, the flash memory write access modifies the data in the memory and the data in the cache.

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

*Note:* The I/D cache must be flushed only when it is disabled (I/DCEN = 0).
3.4 FLASH option bytes

3.4.1 Option bytes 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: Option bytes programming).

A double word is split up in option bytes as indicated in Table 8.

Table 8. Option bytes 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 9. The option bytes can be read from the memory locations listed in Table 9 or from the Option byte registers:

- Flash memory option register (FLASH_OPTR)
- Flash memory PCROP zone A start address register (FLASH_PCROP1ASR)
- Flash memory PCROP zone A end address register (FLASH_PCROP1AER)
- Flash memory PCROP zone B start address register (FLASH_PCROP1BSR)
- Flash memory PCROP zone B end address register (FLASH_PCROP1BER)
- Flash memory WRP area A address register (FLASH_WRP1AR)
- Flash memory WRP area B address register (FLASH_WRP1BR)
- Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR)
- Secure flash memory start address register (FLASH_SFR)
- Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)

Table 9. Option bytes organization

| Address(1) | 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 |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1FFF8000   | AGC_TRIM | Unused | nBOOT0 | nSWBOOT0 | SRAM2_RST | nRST_STOP | Unused | nRST_SHDW | BOR_LEV | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused |
| 1FFF8008   | Unused | PCROP1A_STRT |
| 1FFF8010   | PCROP RDP | Unused | PCROP1A_END |
| 1FFF8018   | Unused | WRP1A_END | Unused | WRP1A_STRT |
| 1FFF8020   | Unused | WRP1B_END | Unused | WRP1B_STRT |

---

(1) Address in hexadecimal format.
### Table 9. Option bytes organization (continued)

<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>
</tr>
</thead>
<tbody>
<tr>
<td>1FFF8028</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1FFF8030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1FFF8038 to 1FFF8060</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1FFF8068</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1FFF8070</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1FFF8078</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1FFF8FF8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>

1. The upper 32-bits of the double-word address contain the inverted data from the lower 32 bits.
2. The value 0x0BC0DE1D is written during production, when option bytes are configured.

### User and read protection option bytes

Flash memory address: 0x1FFF 8000
Reset value: 0b001x 1101 1xxx 1111 x0001 1010 1010 (ST production value)

<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>
</tr>
</thead>
<tbody>
<tr>
<td>AGC_TRIM[2:0]</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>
<td></td>
<td></td>
</tr>
<tr>
<td>Res.</td>
<td>n BOOT0</td>
<td>nSW BOOT0</td>
<td>SRAM2 _RST</td>
<td>SRAM2 _PE</td>
<td>n BOOT1</td>
<td>Res.</td>
<td>Res.</td>
<td>WWDG _SW</td>
<td>WWDG _STDBY</td>
<td>WWDG _STOP</td>
<td>WWDG _SW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>r</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</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></td>
</tr>
<tr>
<td>Res.</td>
<td>nRST _SHDW</td>
<td>nRST _STDY</td>
<td>nRST _STOP</td>
<td>BOR _LEV[2:0]</td>
<td>ESE</td>
<td>RDP[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>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29 **AGC_TRIM[2:0]**: Automatic gain control trimming.
Default value 0b001.

Bit 28 Reserved, must be kept at reset value.

Bit 27 **nBOOT0**: nBOOT0 option bit
0: nBOOT0 = 0
1: nBOOT0 = 1

Bit 26 **nSWBOOT0**: Software BOOT0
0: BOOT0 taken from the option bit nBOOT0
1: BOOT0 taken from PH3/BOOT0 pin

Bit 25 **SRAM2 _RST**: SRAM2 and PKA RAM erase when system reset
0: SRAM2 and PKA RAM erased when a system reset occurs
1: SRAM2 and non-secure PKA RAM not erased when a system reset occurs

Bit 24 **SRAM2 _PE**: SRAM2 parity check enable
0: SRAM2 parity check enabled
1: SRAM2 parity check disabled
Bit 23  nBOOT1: Boot configuration
Together with the BOOT0 pin or option bit nBOOT0 (depending on nSWBOOT0 option bit configuration), this bit selects boot mode from the flash main memory, SRAM1 or the System memory. Refer to Section 2.3: Boot configuration.

Bits 22:20  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  IWDG_SW: Independent watchdog selection
  0: Hardware independent watchdog
  1: Software independent watchdog

Bit 15  Reserved, must be kept at reset value.
Bit 14  nRST_SHDW
  0: Reset generated when entering the Shutdown mode
  1: No reset generated when entering the Shutdown mode

Bit 13  nRST_STDBY
  0: Reset generated when entering the Standby mode
  1: No reset generated when entering the Standby mode

Bit 12  nRST_STOP
  0: Reset generated when entering the Stop mode
  1: No reset generated when entering the Stop mode

Bits 11:9  BOR_LEV[2:0]: BOR reset Level
These bits contain the VDD supply level threshold that activates/releases the reset.
  000: BOR Level 0. Reset level threshold is ~ 1.7 V
  001: BOR Level 1. Reset level threshold is ~ 2.0 V
  010: BOR Level 2. Reset level threshold is ~ 2.2 V
  011: BOR Level 3. Reset level threshold is ~ 2.5 V
  100: BOR Level 4. Reset level threshold is ~ 2.8 V

Bit 8  ESE: System security enabled flag.
  0: Security disabled
  1: Security enabled

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 8008
Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxx1 1111 1111, (STM32WB55), 0xXXXX XXFF (STM32WB35), (ST production value)

PCROP1A end address option bytes

Flash memory address: 0x1FFF 8010
Reset value: 0b1xxx xxxx xxxx xxxx xxxx 0000 0000 (STM32WB55), 0b1xxx xxxx xxxx xxxx xxxx 0000 0000 (STM32WB35) (ST production value)

PCROP1A area start offset
PCROP1A_STRT contains the first 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.

Bit 31 PCROP1A_STRT[8:0]: PCROP1A area start offset
This bit is set only. It is reset after a full mass erase due to a change of RDP from Level 1 to Level 0.
0: PCROP area is not erased when the RDP level is decreased from Level 1 to Level 0.
1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:9 Reserved, must be kept at reset value.

Bits 8:0 PCROP1A_END: PCROP1A area end offset
PCROP1A_END contains the last 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.

WRP Area A address option bytes
Flash memory address: 0x1FFF 8018
Reset value: 0xXX00 XXFF (STM32WB55) 0bxxxx xxxx x000 0000 xxxx xxxx x111 1111 (STM32WB35) (ST production value)
Embedded flash memory (FLASH)

WRP Area A address option bytes
Flash memory address: 0x1FFF 8020
Reset value: 0xXX00 XXFF (STM32WB55), 0bxxxx xxxx x000 0000 xxxx xxxx x111 1111 (STM32WB35) (ST production value)

Bits 31:24 Reserved, must be kept at reset value.
Bits 23:16 **WRP1A_END[7:0]**: WRP area “A” end offset
WRPA1_END contains the last 4 Kbytes page of the WRP area “A”. Note that bit 23 is reserved on STM32WB35xx devices.

Bits 15:8 Reserved, must be kept at reset value.
Bits 7:0 **WRP1A_STRT[7:0]**: WRP area “A” start offset
WRPA1_STRT contains the first 4 Kbytes page of the WRP area “A”. Note that bit 7 is reserved on STM32WB35xx devices.

WRP Area B address option bytes
Flash memory address: 0x1FFF 8020
Reset value: 0xXX00 XXFF (STM32WB55), 0bxxxx xxxx x000 0000 xxxx xxxx x111 1111 (STM32WB35) (ST production value)

Bits 31:24 Reserved, must be kept at reset value.
Bits 23:16 **WRP1B_END[7:0]**: WRP area “B” end offset
WRPB1_END contains the last 4 Kbytes page of the WRP area “B”. Note that bit 23 is reserved on STM32WB35xx devices.

Bits 15:8 Reserved, must be kept at reset value.
Bits 7:0 **WRP1B_STRT[7:0]**: WRP area “B” start offset
WRPB1_STRT contains the first 4 Kbytes page of the WRP area “B”. Note that bit 7 is reserved on STM32WB35xx devices.

PCROP1B start address option bytes
Flash memory address: 0x1FFF 8028
Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx1 1111 1111 (STM32WB55), 0xXXXX XXFF (STM32WB35) (ST production value)
Embedded flash memory (FLASH)  RM0434

PCROP1B end address option bytes

Flash memory address: 0x1FFF 8030
Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx (STM32WB55), 0xXXXX XX00 (STM32WB35) (ST production value)

<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_END[8:0]

Bits 31:9 Reserved, must be kept at reset value.
Bits 8:0 PCROP1B_END[8:0]: PCROP1B area end offset
PCROP1B_END contains the last 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB35xx devices.

IPCC mailbox data buffer address option bytes

Flash memory address: 0x1FFF 8068
Reset value: 0bXXXX XXXX XXXX XXXX XXXX 0000 0000 0000 0000 0000 0000 0000 0000 (ST production value)

<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>

IPCCDBA[13:0]

Bits 31:14 Reserved, must be kept at reset value.
Bits 13:0 IPCCDBA[13:0]: IPCC mailbox data buffer base address offset
IPCCDBA contains the first double-word offset of the IPCC mailbox data buffer area in SRAM2.
### Secure flash memory start address option bytes

Flash memory address: 0x1FFF 8070

Reset value: 0bxxxx xxxx xxxx xxxx1 xxx0 xxxx xxxx (ST production value)

<table>
<thead>
<tr>
<th>Bits 31:13</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 12</td>
<td><strong>DDS</strong>: Disable CPU2 debug access</td>
</tr>
<tr>
<td></td>
<td>0: CPU2 debug access enabled.</td>
</tr>
<tr>
<td></td>
<td>1: CPU2 debug access disabled.</td>
</tr>
<tr>
<td>Bits 11:9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>FSD</strong>: Flash security disable</td>
</tr>
<tr>
<td></td>
<td>FSD = 1: System and flash non-secure</td>
</tr>
<tr>
<td></td>
<td>FSD = 0: System and flash secure.</td>
</tr>
<tr>
<td></td>
<td>SFSA[7:0] contains the start address of the first 4 Kbytes page of the secure flash area.</td>
</tr>
<tr>
<td>Bits 7:0</td>
<td><strong>SFSA[7:0]</strong>: Secure flash start address</td>
</tr>
<tr>
<td></td>
<td>System and flash secure. SFSA[7:0] contains the start address of the first 4 Kbytes page of the secure flash area.</td>
</tr>
</tbody>
</table>

### Secure SRAM2 start address and CPU2 reset vector option bytes

Flash memory address: 0x1FFF 8078

Reset value: 0xXXXX XXXX (ST production value)

| Bits 31     | **C2OPT**: CPU2 boot reset vector memory selection. |
|            | 0: SBRV offset addresses SRAM1 or SRAM2, from start address 0x2000 0000 (SBRV value must be kept within the SRAM area). |
|            | 1: SBRV offset addresses flash memory, from start address 0x0800 0000. |
| Bit 30     | **NBRSD**: Non-backup SRAM2b security disable |
|            | NBRSD = 1: SRAM2b is non-secure. |
|            | NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area. |
| Bits 29:25 | **SNBRSA**: Secure non-backup SRAM2b start address |
|            | NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area. |
3.4.2 Option bytes programming

After reset, the options related bits in the Flash memory control register (FLASH_CR) and Flash memory CPU2 control register (FLASH_C2CR) are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in the Flash memory 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 in the Flash memory option key register (FLASH_OPTKEYR)
3. Write OPTKEY2 = 0x4C5D6E7F in the Flash memory option key register (FLASH_OPTKEYR)

Any wrong sequence locks 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.

Note: In a multi-CPU system it is good practice to use semaphores to manage option programming, and prevent simultaneous option programming by the CPUs.

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 options value in the options registers.
3. Check that no flash memory operation is on going by checking the BSY bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR).

Check that flash program and erase operation is allowed by checking the PESD bit in the Flash memory status register (FLASH_SR) or Flash memory CPU2 status register (FLASH_C2SR) (these checks are recommended even if status may change due to

---

Bit 24 Reserved, must be kept at reset value.

Bit 23 **BRSD**: Backup SRAM2a security disable
  - BRSD = 1: SRAM2a is non-secure
  - BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 22:18 **SBRSA**: Secure backup SRAM2a start address
  - BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 17:0 **SBRV[17:0]**: CPU2 boot reset vector
  - Contains the word aligned CPU2 boot reset start address offset within the selected memory area by C2OPT.
flash operation requests by the other CPU, to limit the risk of receiving a bus error when modifying user options).

4. Set the Options start bit OPTSTRT in the Flash memory control register (FLASH_CR).
5. Wait for the BSY bit to be cleared.

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

### Secure user options

The secure option bytes Secure flash memory start address register (FLASH_SFR) and Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR) can only be written by the secure CPU2.

### Option byte loading

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

Option bytes loading is performed in two cases:
- when OBL_LAUNCH bit is set in the Flash memory control register (FLASH_CR)
- 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 all options at ‘1’, except for BOR_LEV that is “000” (lowest threshold)
- 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”
- for FSD and SFSA option, the value of mismatch is “all memories (Flash, SRAM2a and SRAM2b) secured”
- for BRSD, SBRSA, NBRSD and SNBRASA options, the value of mismatch is “SRAM2 memory part secured”
- for DDS option, the value of mismatch is “CPU2 debug disabled”
- for C2OPT and SBRV options, the value of mismatch is “CPU2 boot from start address of last Flash page (safe boot)”
- for OPTVAL option, the value of mismatch is “not valid”.

If the OPTVAL option indicates “not valid” all memories (Flash, SRAM2a and SRAM2b) are secure.

### Table 10. Option loading control

<table>
<thead>
<tr>
<th>OPTVERR</th>
<th>OPTNV</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Options correctly loaded and OPTVAL is “valid”.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Does not occur</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>OPTVAL option correctly loaded as “valid”, but some or all other option and engineering bits corrupted, mismatch values loaded.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Some or all option and engineering bits corrupted, mismatch values loaded. OPTVAL correctly loaded as “not valid”. Security applied on full memories irrespective of the loaded secure option values.</td>
</tr>
</tbody>
</table>

On system reset rising, internal option registers are copied into option registers that can be read and written by software:

- `FLASH.OPTR`
- `FLASH_PCROPxySR (x = 1 and y = A or B)`
- `FLASH_PCROPxyER (x = 1 and y = A or B)`
- `FLASH_WRPxyR (x = 1 and y = A or B)`
- `FLASH_IPCCDBA`
- `FLASH_SFR`
- `FLASH_SRRVA`

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 UID64**

A 64-bit unique device identification (UID64) is stored in the flash memory, and can be accessed by the CPUs.

| Address   | 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  |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x1FFF 7580 | 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  | Unique device number |
| 0x1FFF 7584 | 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  | ST company ID |

The UID64 is programmed at device production, and provides a unique code for each device:
- the 24-bit ST company ID returns 0x00 80 E1
- the 8-bit device ID returns 0x26 for STM32WB55xx revision Y and STM32WB35xx revision A, 0x27 for STM32WB55xx/35xx revision X, as indicated in DBGMCU_IDCODE
- the 32-bit unique device number is a sequential number, different for each individual device.
3.6  Flash memory protection

The flash main 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 4 Kbytes. Apart from the RDP and WRP, flash memory can also be protected against read and write from third parties (PCROP). The PCROP granularity is 2 Kbytes.

Part of the flash main memory can be secured. It grants exclusive access to this part of the memory to the CPU2.

3.6.1  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 flash main memory, the option bytes, the backup registers (RTC_BKPxR in the RTC) and the SRAM2.

**Note:** If the read protection is set while the debugger is still connected (or was connected since the last power on) through JTAG/SWD, apply a POR (power-on reset) instead of a system reset. If the read protection is programmed through software, do not set the OBL_LAUNCH bit (FLASH_CR register), but perform a POR to reload the option byte. This can be done with a transition to Standby (or Shutdown) mode, followed by a wake-up.

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.

<table>
<thead>
<tr>
<th>RDP byte value</th>
<th>RDP complement 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 value except 0xAA or 0xCC</td>
<td>Any value (not necessarily complementary), except 0x55 and 0x33</td>
<td>Level 1 (default)</td>
</tr>
<tr>
<td>0xCC</td>
<td>0x33</td>
<td>Level 2</td>
</tr>
</tbody>
</table>

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 into the flash main memory area are possible. The option bytes, the SRAM2 and the backup registers are also accessible by all operations.
Level 1: Read protection

This is the default protection level when RDP option byte is erased. It is defined as well when RDP value is at any value different from 0xAA and 0xCC, or even if the complement is not correct.

**User mode:** Code executing in user mode *(Boot flash)* can access flash main memory, option bytes, SRAM2 and backup registers with all operations.

**Debug, boot RAM and boot loader modes:** In debug mode or when code is running from boot RAM or boot loader, the flash main memory, the backup registers *(RTC_BKPxR in the RTC)* and the SRAM2 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 case the Level 1 is configured and no PCROP areas are 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 case the Level 1 is configured and a PCROP area is defined, if user code needs to be protected by RDP but not by PCROP, it must not be placed in a page containing a PCROP area.

Level 2: No debug

In this level, the protection Level 1 is guaranteed. In addition, the CPU1 and CPU2 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 flash main memory. On the contrary, only read and secure write operations can be performed on the option bytes. Option bytes, can only be programmed and erased by a secure CPU2.

The Level 2 cannot be removed from the non-secure application side: it is an irreversible operation. When attempting to modify the options bytes, the protection error flag WRPERR is set in the FLASH_xxxSR register and an interrupt can be generated.

**Note:** The debug feature 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

It is easy to move from Level 0 to Level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from Level 0 or from Level 1. Once in Level 2 it is no more possible to modify the Read protection level.

When the RDP is reprogrammed to the value 0xAA to move from Level 1 to Level 0, a mass erase of the flash main memory is performed if PCROP_RDP is set in the *Flash memory PCROP zone A end address register (FLASH_PCROP1AER)*. The backup registers *(RTC_BKPxR in the RTC)*, the SRAM2 and the PKA SRAM are also erased. The user options (except PCROP protection) are set to their previous values copied from FLASH_OPTR, FLASH_WRPxyR *(x = 1 and y = A or B)*. PCROP is disabled. The OTP area is not affected by mass erase and remains unchanged.

If the bit PCROP_RDP is cleared in the FLASH_PCROP1AER, the full mass erase is replaced by a partial mass erase that is successive page erases, except for the pages protected by PCROP. This is done in order to keep the PCROP code. Only when the flash...
memory is erased, options are re-programmed with their previous values. This is also true for FLASH_PCROPxySR and FLASH_PCROPxyER registers (x = 1 and y = A or B).

A requested mass erase performs a partial mass erase (a sequence of page erases), except for the pages protected by CPU2 security (SFSA). This is done to keep the CPU2 secure code.

Table 13. RDP regression from Level 1 to Level 0 and memory erase

<table>
<thead>
<tr>
<th>SFSA</th>
<th>PCROP</th>
<th>PCROP_RDP</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial</td>
<td>None</td>
<td>x</td>
<td>Flash memory multiple page erase of all non-secure pages, SRAM2, PKA RAM, and backup registers erase (secure flash pages conserved).</td>
</tr>
<tr>
<td>Partial</td>
<td>1</td>
<td></td>
<td>Flash memory multiple page erase of all non-PCROP pages and non-secure pages, SRAM2, PKA RAM, and backup registers erase (PCROP flash memory pages and secure flash memory pages conserved).</td>
</tr>
<tr>
<td>Complete non-secure</td>
<td>0</td>
<td></td>
<td>Flash memory, SRAM2, and PKA RAM and backup registers are conserved.</td>
</tr>
<tr>
<td>Complete flash memory</td>
<td>x</td>
<td>x</td>
<td>Flash memory, SRAM2, and PKA RAM and backup registers are conserved.</td>
</tr>
</tbody>
</table>

Note: Partial mass erase is performed only when Level 1 is active and Level 0 requested. When the protection level is increased (0→1, 1→2, 0→2) there is no mass erase.

To validate the protection level change, the option bytes must be reloaded through the OBL_LAUNCH bit in flash memory control register.
Figure 4. Changing the Read protection (RDP) level

Table 14. Access status vs. protection level and execution modes

<table>
<thead>
<tr>
<th>Area</th>
<th>Protection level</th>
<th>User execution (BootFromFlash memory)</th>
<th>Debug / BootFromRam / BootFromLoader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read</td>
<td>Write</td>
<td>Erase</td>
</tr>
</tbody>
</table>
| Flash main memory  | 1 Yes | Yes | Yes | No | No | No
|                    | 2 Yes | Yes | Yes | N/A(1) | N/A(1) | N/A(1) |
| System memory(2)   | 1 Yes | No | No | Yes | No | No |
|                    | 2 Yes | No | No | N/A(1) | N/A(1) | N/A(1) |
| Option bytes       | 1 Yes | Yes(3) | Yes | Yes | Yes(3) | Yes |
|                    | 2 Yes | CPU1 and CPU2 none secure - No | CPU1 and CPU2 none secure - No | N/A(1) | N/A(1) | N/A(1) |
|                    |     | CPU2 secure - Yes | CPU2 secure - Yes | N/A(1) | N/A(1) | N/A(1) |
| Backup registers   | 1 Yes | Yes | N/A | No | No | No(4) |
|                    | 2 Yes | Yes | N/A | N/A(1) | N/A(1) | N/A(1) |
| SRAM2              | 1 Yes | Yes | N/A | No | No | No(5) |
|                    | 2 Yes | Yes | N/A | N/A(1) | N/A(1) | N/A(1) |

1. When the protection Level 2 is active, the Debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3.6.2 Proprietary code readout protection (PCROP)

Two parts of the flash memory can be protected against read and write from third parties. The protected area is execute-only: it can only be reached by the STM32 CPUs, with an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP areas have a 2 Kbytes granularity. An additional option bit (PCROP_RDP) makes it possible 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 page offset and an end page offset into the flash memory. These offsets are defined in the PCROP address registers Flash memory PCROP zone A start address register (FLASH_PCROP1ASR), Flash memory PCROP zone A end address register (FLASH_PCROP1AER), Flash memory PCROP zone B start address register (FLASH_PCROP1BSR) and Flash memory PCROP zone B end address register (FLASH_PCROP1BER).

A PCROP area is defined from the address Flash memory Base address + [PCROPxy_STRT x 0x800] (included) to the address: Flash memory Base address + [(PCROPxy_END+1) x 0x800] (excluded). The minimum PCROP area size is two PCROP pages (4 Kbytes) PCROPxy_END = PCROPxy_STRT + 1.

When PCROPxy_END = PCROPxy_STRT the full flash memory is PCROP protected.

For example, to protect by PCROP from the address 0x0806 2F80 (included) to the address 0x0807 0004 (included):

- if boot in flash memory is selected, one of the FLASH_PCROPxySR and FLASH_PCROPxyER registers (x = 1 and y = A or B) must be programmed with:
  - PCROPxy_STRT = 0xC5 (PCROP area first address 0x0806 2800)
  - PCROPxy_END = 0xE0 (PCROP area last address 0x0807 07FF)

Any data read access performed through a PCROP protected area triggers the RDERR flag error.

Any PCROP protected address is also write protected and any write access to one of these addresses triggers WRPERR.

Any PCROP area is also erase protected. Consequently, any erase to a page in this zone is impossible (including the page containing the start address and the end address of this zone). Moreover, a software mass erase cannot be performed if one zone is PCROP protected.

In the previous example, due to erase by page, all pages from page 0xC5 to 0xE0 are protected in case of page erase, all addresses from 0x0806 2800 to 0x0807 07FF cannot be erased.

Deactivation of PCROP can only occur when the RDP is changing from Level 1 to Level 0. If the user options modification tries to clear PCROP or to decrease the PCROP areas, the
options programming is launched but PCROP areas stays unchanged. On the contrary, it is possible to increase the PCROP areas.

When option bit PCROP_RDP is cleared, and when the RDP is changing from Level 1 to Level 0, Full mass erase is replaced by Partial mass erase to preserve the PCROP area (refer to Changing the Read protection level). In this case, PCROPxy_STRT and PCROPxy_END \((x = 1 \text{ and } y = A \text{ or } B)\) are not erased.

### Table 15: PCROP protection

<table>
<thead>
<tr>
<th>PCROP registers values ((x = 1 \text{ and } y = A \text{ or } B))</th>
<th>PCROP protection area</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCROPxy_STRT = PCROPxy_END</td>
<td>The full flash memory is PCROP protected</td>
</tr>
<tr>
<td>PCROPxy_STRT &gt; PCROPxy_END</td>
<td>No PCROPxy, unprotected</td>
</tr>
<tr>
<td>PCROPxy_STRT &lt; PCROPxy_END</td>
<td>Pages from PCROPxy_STRT to PCROPxy_END are protected</td>
</tr>
</tbody>
</table>

**Note:** It is recommended to align PCROP areas with page granularity when using PCROP_RDP, or to leave free the rest of the page where PCROP zones start or end.

#### 3.6.3 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 (4 Kbytes) 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 memory WRP area A address register (FLASH_WRP1AR) and Flash memory WRP area B address register (FLASH_WRP1BR).

The WRP “y” area \((y = A, B)\) is defined from the address Flash memory Base address + \([\text{WRP1y}\_\text{STRT } x 0x1000]\) (included) to the address Flash memory Base address + \([\text{WRP1y}\_\text{END}+1) x 0x1000]\) (excluded). The minimum WRP area size is one WRP page (4 Kbytes), \(\text{WRP1y}\_\text{END} = \text{WRP1y}\_\text{STRT}\).

For example, to protect by WRP from the address 0x0806 2000 (included) to the address 0x0807 1FFF (included):

- If boot in flash memory is selected, FLASH_WRP1AR register must be programmed with:
  - \(\text{WRP1A}\_\text{STRT} = 0x62\).
  - \(\text{WRP1A}\_\text{END} = 0x71\).

- \(\text{WRP1B}\_\text{STRT}\) and \(\text{WRP1B}\_\text{END}\) 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) is set in the FLASH_SR register. 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.
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 (JTAG or single wire) or boot code is being executed from RAM or system flash memory, even if WRP is not activated. Any attempt generates an hard fault (BusFault).

Table 16: WRP protection

<table>
<thead>
<tr>
<th>WRP registers values (x = 1 and y = A or B)</th>
<th>WRP protection area</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRPxy_STRT = WRPxy_END</td>
<td>Page WRPxy is protected</td>
</tr>
<tr>
<td>WRPxy_STRT &gt; WRPxy_END</td>
<td>No WRP, unprotected</td>
</tr>
<tr>
<td>WRPxy_STRT &lt; WRPxy_END</td>
<td>Pages from WRPxy_STRT to WRPxy_END are protected</td>
</tr>
</tbody>
</table>

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

3.6.4 CPU2 security (ESE)

All or a part of the flash memory and the SRAM2a and SRAM2b can be made secure, exclusively accessible by the CPU2, protected against execution, read and write from third parties. Only the CPU2 can execute, read and write in these areas. It can only be reached by the CPU2, while all other accesses (CPU1 and DMA) are strictly prohibited.

Changing the CPU2 security mode

CPU2 security start address can be modified by the secure CPU2 by loading a new user option SFSA.

CPU2 secure flash memory area

The CPU2 secure flash memory area has sector (4 Kbytes) granularity and is defined by the secure flash memory start page offset user option (SFSA) into the flash memory. This offset is controlled from the SFSA field in the Secure flash memory start address register (FLASH_SFR).

The CPU2 secure flash memory area is defined as follows: Flash memory Base address + [SFSA x 0x1000] (included) to the last flash memory address: When CPU2 security is enabled, the minimum CPU2 secure area size is one sector (4 Kbytes).

For example, a CPU2 secure area from the address 0x080E 7000 (included) to the address 0x080F FFFF (included):

- FLASH_SFR registers must be programmed with:
  - SFSA= 0x0E7.

A flag (ESE) is available from the Flash memory option register (FLASH_OPTR) informing that CPU2 security is enabled.

Any CPU1 access to a CPU2 security area triggers RDERR or WRPERR flag error.

CPU2 secure SRAM2 areas

The CPU2 secure SRAM2a and SRAM2b areas have a 1 Kbyte granularity and are defined by the secure backup RAM (SRAM2a) start address user options (BRSD and SBRSA) and the secure non-backup RAM (SRAM2b) start address user options (NBRSD and SNBRSA) into the flash memory. These offset are controlled by the SBRSA and SNBRSA fields in the
**Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)**

The CPU2 secure SRAM2a area is defined as Backup SRAM2a Base address + \([SBRSA x 0x0400]\) (included) to the last SRAM2a address.

As an example, for a CPU2 secure SRAM2a area from the address 0x2003 5000 (included) to the address 0x2003 7FFF (included) FLASH_SRRVR registers must be programmed with SBRSA = 0x14.

Any CPU1 read access returns no data, and a write access to a CPU2 security SRAM2a area is discarded and triggers a bus error.

When BRSD is set to 1 the SRAM2a is non-secure.

The CPU2 secure non-backup SRAM2b area is defined as Non-backup SRAM2b base address + \([SNBRS A x 0x0400]\) (included) to the last SRAM2b address.

As an example, for a CPU2 secure SRAM2b area from the address 0x2003 EC00 (included) to the address 0x2003 FFFF (included) FLASH_SRRVR registers must be programmed with SNBRSA = 0x1B.

Any CPU1 read access returns no data, and a write access to a CPU2 security SRAM2b area is discarded and triggers a bus error.

When NBRSD is set to 1 the SRAM2b is non-secure.

**CPU2 debug access**

Debug access to the CPU2 is disabled, as indicated by the DDS filed in the Secure flash memory start address register (FLASH_SFR). The debugger has no access to the CPU2 and the secure peripherals and memory areas.

### 3.7 FLASH program/erase suspension

Flash memory program and erase operations can be suspended by setting the PES bit in the Flash memory access control register (FLASH_ACR) or Flash memory CPU2 access control register (FLASH_C2ACR). This feature is useful when executing time critical sections by a CPU. It makes it possible to suspend any new program or erase operation from being started, preventing CPU instruction and data fetches from being blocked.

When at least one PES bit is set:

- Any ongoing program or erase operation is completed.
  - The maximum latency for a flash memory program erase suspension is the maximum time for one program or erase operation to complete (see the device datasheet for more information on the program and erase timing).
  - All new requested program and erase operations are not started, but suspended.
  - PESD bits in the Flash memory status register (FLASH_SR) and Flash memory CPU2 status register (FLASH_C2SR) are set as soon as any PES is set, no matter if a program/erase is currently suspended. This allows a CPU to test PESD prior to requesting a program or an erase operation.
When all PES bits are reset to 0:

- A suspended program or erase operation is started.

  - The PESD bit in both the Flash memory status register (FLASH_SR) and in the Flash memory CPU2 status register (FLASH_C2SR) are cleared.

### 3.8 FLASH interrupts

**Table 17. Flash memory interrupt requests**

<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(^{(1)})</td>
<td>Write EOP = 1</td>
<td>EOPIE</td>
</tr>
<tr>
<td>Operation error</td>
<td>OPER(^{(2)})</td>
<td>Write OPER = 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>
<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>ECC(^{\text{c}})</td>
<td>Write ECC(^{\text{c}}) = 1</td>
<td>ECCCIE</td>
</tr>
<tr>
<td>ECC double error (NMI)</td>
<td>ECCD(^{\text{c}})</td>
<td>Write ECCD(^{\text{c}}) = 1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. EOP is set only if EOPIE is set.
2. OPER is set only if ERRIE is set.

*Note:* The flash interface provides only a single interrupt line to both CPUs. It is good practice to manage flash operations between the CPUs using a semaphore. To avoid receiving unwanted flash operation interrupts, the CPU must mask the flash interrupt in the NVIC or SYSCFG pre-mask.

### 3.9 Register access protection

The user options registers can be protected by security.

The FLASH secure registers (FSD, SFSA, BRSD, SBRSA, NBRSD, SNBRSA, SBRV, C2OPT, and DDS) are secure and can be written only by the secure CPU2 and read by any CPU, secure and non-secure. When the CPU1 tries to write, the write is discarded.
3.10 FLASH registers

3.10.1 Flash memory access control register (FLASH_ACR)

Address offset: 0x000
Reset value: 0x0000 0600

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td><strong>EMPTY</strong>: Flash memory user area empty.</td>
<td>0: Read: User flash memory programmed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Read: User flash memory empty</td>
</tr>
<tr>
<td></td>
<td>When written this bit is overwitten with the written value.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td><strong>PES</strong>: CPU1 program / erase suspend request</td>
<td>0: Flash memory program and erase operations granted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Any new flash memory program and erase operation is suspended until this bit and the same bit in Flash memory CPU2 access control register (FLASH_C2ACR) are cleared. The PESD bit in both the Flash memory status register (FLASH_SR) and Flash memory CPU2 status register (FLASH_C2SR) is set when at least one PES bit in FLASH_ACR or FLASH_C2ACR is set.</td>
</tr>
<tr>
<td>14:13</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td><strong>DCRST</strong>: CPU1 Data cache reset</td>
<td>0: CPU1 Data cache is not reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CPU1 Data cache is reset</td>
</tr>
<tr>
<td></td>
<td>This bit can be written only when the data cache is disabled.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td><strong>ICRST</strong>: CPU1 Instruction cache reset</td>
<td>0: CPU1 Instruction cache is not reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CPU1 Instruction cache is reset</td>
</tr>
<tr>
<td></td>
<td>This bit can be written only when the instruction cache is disabled.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><strong>DCEN</strong>: CPU1 Data cache enable</td>
<td>0: CPU1 Data cache is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CPU1 Data cache is enabled</td>
</tr>
<tr>
<td>9</td>
<td><strong>ICEN</strong>: CPU1 Instruction cache enable</td>
<td>0: CPU1 Instruction cache is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CPU1 Instruction cache is enabled</td>
</tr>
<tr>
<td>8</td>
<td><strong>PRFTEN</strong>: CPU1 Prefetch enable</td>
<td>0: CPU1 Prefetch disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CPU1 Prefetch enabled</td>
</tr>
</tbody>
</table>

Bits 31:17 Reserved, must be kept at reset value.
Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 \textbf{LATENCY[2:0]}: Latency
These bits represent the ratio of the flash memory HCLK clock period to the flash memory access time.
- 000: Zero wait states
- 001: One wait state
- 010: Two wait states
- 011: Three wait states
- Others: Reserved

3.10.2 Flash memory key register (FLASH_KEYR)

Address offset: 0x008
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>
<th>KEY[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0 \textbf{KEY[31:0]}: Flash memory key
The following values must be written consecutively to unlock the Flash memory control register (FLASH_CR) and Flash memory CPU2 control register (FLASH_C2CR), thus enabling programming/erasing operations:
- KEY1: 0x4567 0123
- KEY2: 0xCDEF 89AB

3.10.3 Flash memory option key register (FLASH_OPTKEYR)

Address offset: 0x00C
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>
<th>OPTKEY[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
<td>w w w w w w w w w w w w w w w w w w w w w w w</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0 \textbf{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
3.10.4 Flash memory status register (FLASH_SR)

Address offset: 0x010
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>PESD: Programming/erase operation suspended</td>
</tr>
<tr>
<td>18</td>
<td>CFGBSY: Programming or erase configuration busy</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>BSY: Busy</td>
</tr>
<tr>
<td>15</td>
<td>OPTVERR: Option and Engineering bits loading validity error</td>
</tr>
<tr>
<td>14</td>
<td>RDERR: PCROP read error</td>
</tr>
<tr>
<td>13</td>
<td>OPTNV: User option OPTVAL indication</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>

Bit 19 PESD: Programming/erase operation suspended
This bit is set and reset by hardware.
Set when at least one PES bit in either Flash memory access control register (FLASH_ACR) or Flash memory CPU2 access control register (FLASH_C2ACR) is set.
Cleared when both PES bits in FLASH_ACR and FLASH_C2ACR are cleared.
When set, new program or erase operations are not started.

Bit 18 CFGBSY: Programming or erase configuration busy
This flag is set and reset by hardware:
- set when first word is sent for program, or when setting STRT for erase
- reset when operation completes or is interrupted by an error.
When set to 1, any other operation launch through Flash memory control register (FLASH_CR) is impossible and must be postponed (a programming or erase operation is ongoing). When reset to 0, programming and erase settings in Flash memory control register (FLASH_CR) can be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 BSY: Busy
Indicates that a flash memory operation requested by Flash memory 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 have not been properly loaded, OPTVERR is set again after each system reset. Option bytes that fail loading are forced to a safe value, see Section 3.4.2: Option bytes programming.
Cleared by writing 1.

Bit 14 RDERR: PCROP read error
Set by hardware when an address to be read through the D-bus 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.

Bit 13 OPTNV: User option OPTVAL indication
This bit is set and reset by hardware.
0: The OBL user option OPTVAL indicates “valid”.
1: The OBL user option OPTVAL indicates “not valid”.

Bits 12:10 Reserved, must be kept at reset value.
3.10.5 Flash memory 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 must not be modified when CFGBSY in Flash memory status register (FLASH_SR) is set. This would result in:

- a bus error, if CFGBSY is set due to a double-word programming sequence and the write to FLASH_CR is performed after the first word is sent for programming and before the second.
- a bus stall, if CFGBSY is set for any other reason
  - When the PESD bit in Flash memory status register (FLASH_SR) is cleared, the register write access is stalled until the CFGBSY bit is cleared. (by the other CPU)
  - When the PESD bit in Flash memory status register (FLASH_SR) is set and a program or an erase operation is ongoing, the register write access causes a bus error.
  - When the PESD bit in Flash memory status register (FLASH_SR) is set, but there is no ongoing programming or erase operation, the register write access is completed. The requested program or erase operation is suspended, the BSY/CFGBSY asserted and remains 1, until suspend is deactivated. Consequently, PESD bit goes back to 0 and the suspended operation completes.

<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>rs</td>
<td>rs</td>
<td>rs</td>
<td>rc_w1</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</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>

Bit 31 **LOCK**: FLASH_CR lock
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.

Bit 30 **OPTLOCK**: Options lock
This bit is set only. When set, all bits concerning 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.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **OBL_LAUNCH**: Forces the option byte loading
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.
0: Option byte loading complete
1: Option byte loading requested

Bit 26 **RDERRIE**: PCROP read error interrupt enable
This bit enables the interrupt generation when the RDERR bit in the FLASH_SR is set to 1.
0: PCROP read error interrupt disabled
1: PCROP read error interrupt enabled
### 3.10.6 Flash memory 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>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 in the FLASH_SR is set to 1.</td>
</tr>
<tr>
<td>0</td>
<td>OPERR error interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>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 in the FLASH_SR is set to 1.</td>
</tr>
<tr>
<td>0</td>
<td>EOP Interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>EOP Interrupt enabled</td>
</tr>
</tbody>
</table>

| Bits 23:19 | Reserved, must be kept at reset value. |

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

<table>
<thead>
<tr>
<th>Bit 17</th>
<th>OPTSTRT: Options modification start</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 BSY bit is cleared in FLASH_SR.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>STRT: Start</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit triggers an erase operation when set. If MER and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition must be forbidden.</td>
</tr>
<tr>
<td></td>
<td>This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_SR. Starting operations by the CPU1, involving secure flash memory pages are rejected and a bus error is generated.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bits 10:3</th>
<th>PNB[7:0]: Page number selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>page 0</td>
</tr>
<tr>
<td>0x01</td>
<td>page 1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0x7F</td>
<td>page 127</td>
</tr>
<tr>
<td>... (STM32WB55xx only)</td>
<td></td>
</tr>
<tr>
<td>0xFF</td>
<td>page 255 (STM32WB55xx only)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>MER: Mass erase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit triggers the mass erase (all user pages) when set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>PER: Page erase</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>page erase disabled</td>
</tr>
<tr>
<td>1</td>
<td>page erase enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>PG: Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Flash memory programming disabled</td>
</tr>
<tr>
<td>1</td>
<td>Flash memory programming enabled</td>
</tr>
</tbody>
</table>

---

**STM32**
3.10.7 Flash memory option register (FLASH_OPTR)

Address offset: 0x020

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx1 xxxx xxxx (the option bits are loaded with values from flash memory at reset release)

Access: no wait state when no flash memory operation is ongoing, word, half-word and byte access.

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

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

### Bit 29 Reserved, must be kept at reset value.

### Bits 28:26 CPUID[2:0]: CPU identification
Set by hardware, indicates the Bus-ID of the CPU access causing the ECC failure.
000: CPU1 access
001: CPU2 access

### Bit 25 Reserved, must be kept at reset value.

### Bit 24 ECCCIE: ECC correction interrupt enable
0: ECCC interrupt disabled
1: ECCC interrupt enabled

### Bits 23:21 Reserved, must be kept at reset value.

### Bit 20 SYSF_ECC: System flash memory ECC fail
This bit indicates that the ECC error correction or double ECC error detection is located in the system flash memory.

### Bits 19:17 Reserved, must be kept at reset value.

### Bits 16:0 ADDR_ECC[16:0]: ECC fail double-word address
These bits indicate that double word address is concerned by the ECC error correction or causes the double ECC error detection.
Bits 31:29 **AGC_TRIM[2:0]**: Radio automatic gain control trimming
   Default value 0b001

   Bit 28 Reserved, must be kept at reset value.

   Bit 27 **nBOOT0**: nBOOT0 option bit
   If nSWBOOT0 bit configuration select BOOT0 is taken from bit nBOOT0, together with bit nBOOT1, selects boot from the user flash memory, SRAM1, or system flash memory. Refer to Section 2.3: Boot configuration.
   0: nBOOT0 = 0
   1: nBOOT0 = 1

   Bit 26 **nSWBOOT0**: Software BOOT0 selection
   0: BOOT0 taken from the option bit nBOOT0
   1: BOOT0 taken from PH3/BOOT0 pin

   Bit 25 **SRAM2_RST**: SRAM2 and PKA RAM Erase when system reset
   0: SRAM2 and PKA RAM erased when a system reset occurs
   1: SRAM2 and non-secure PKA RAM not erased when a system reset occurs

   Bit 24 **SRAM2_PE**: SRAM2 parity check enable
   0: SRAM2 parity check enabled
   1: SRAM2 parity check disabled

   Bit 23 **nBOOT1**: Boot configuration
   Together with the BOOT0 pin or option bit nBOOT0 (depending on nSWBOOT0 option bit configuration), this bit selects boot mode from the user flash memory, SRAM1 or the System memory. Refer to Section 2.3: Boot configuration.

   Bits 22:20 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 **IWDG_SW**: Independent watchdog selection
   0: Hardware independent watchdog
   1: Software independent watchdog

   Bit 15 Reserved, must be kept at reset value.
3.10.8 Flash memory PCROP zone A start address register (FLASH_PCROP1ASR)

Address offset: 0x024

Reset value: 0xXXXX XXXX

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>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[8:0]

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

Bits 8:0 PCROP1A_STRT[8:0]: PCROP1A area start offset

PCROP1A_STRT contains the first 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.
### 3.10.9 Flash memory PCROP zone A end address register (FLASH_PCROP1AER)

Address offset: 0x028

Reset value: 0xXXXX XXXX

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

PCROP_RDP bit can be accessed with 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>rs</td>
<td></td>
<td></td>
<td></td>
<td></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>PCROP1A_END[8:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **PCROP_RDP**: PCROP area preserved when RDP level decreased

This bit is set only. It is reset after a full mass erase due to a change of RDP from Level 1 to Level 0.

0: PCROP area is not erased when the RDP level is decreased from Level 1 to Level 0.

1: PCROP area is erased when the RDP level is decreased from Level 1 to Level 0 (full mass erase).

Bits 30:9 Reserved, must be kept at reset value.

Bits 8:0 **PCROP1A_END[8:0]**: PCROP1A area end offset

PCROP1A_END contains the last 2 Kbytes page of the PCROP1A area. Note that bit 8 is reserved on STM32WB35xx devices.

### 3.10.10 Flash memory WRP area A address register (FLASH_WRP1AR)

Address offset: 0x02C

Reset value: 0xXXXX XXXX

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>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>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>WRP1A_STRT[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRP1A_END[7:0]**: WRP first area "A" end offset

Contains the last 4 Kbytes page of the WRP first area. Note that bit 23 is reserved on STM32WB35xx devices.

Bits 15:8 Reserved, must be kept at reset value.
3.10.11 Flash memory WRP area B address register (FLASH_WRP1BR)

Address offset: 0x030
Reset value: 0xXXXX XXXX
Access: no wait state when no flash memory operation is on going, word, half-word and byte access.

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

Bits 7:24: Reserved, must be kept at reset value.

Bits 23:16 **WRP1B_END[7:0]**: WRP second area “B” end offset
WRP1B_END contains the last 4 Kbytes page of the WRP second area. Note that bit 23 is reserved on STM32WB35xx devices.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **WRP1B_STRT[7:0]**: WRP second area “B” start offset
WRP1B_END contains the first 4 Kbytes page of the WRP second area. Note that bit 7 is reserved on STM32WB35xx devices.

3.10.12 Flash memory PCROP zone B start address register (FLASH_PCROP1BSR)

Address offset: 0x034
Reset value: 0xXXXX XXXX
Access: no wait state when no flash memory operation is on going, word, half-word access.

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

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

Bits 8:0 **PCROP1B_STRT[8:0]**: PCROP1B area start offset
Contains the first 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB35xx devices.
### 3.10.13 Flash memory PCROP zone B end address register (FLASH_PCROP1BER)

Address offset: 0x038  
Reset value: 0xXXXX XXXX  
Access: no wait state when no flash memory operation is on going, word, half-word access.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>PCROP1B_END[8:0]</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>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 8:0  **PCROP1B_END[8:0]**: PCROP1B area end offset  
Contains the last 2 Kbytes page of the PCROP1B area. Note that bit 8 is reserved on STM32WB35xx devices.

### 3.10.14 Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR)

Address offset: 0x03C  
Reset value: 0xXXXX XXXX

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

Bits 31:14  Reserved, must be kept at reset value.

Bits 13:0  **IPCCDBA[13:0]**: IPCC mailbox data buffer base address offset  
Contains the first double-word offset of the IPCC mailbox data buffer area in SRAM2.

### 3.10.15 Flash memory CPU2 access control register (FLASH_C2ACR)

Address offset: 0x05C  
Reset value: 0x0000 0600

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>PES</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>ICRST</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>ICEN</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>PRFTEN</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>rw</td>
</tr>
</tbody>
</table>
3.10.16 Flash memory CPU2 status register (FLASH_C2SR)

Address offset: 0x060
Reset value: 0x0000
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>CFGBSY</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>RSY</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>BSY</td>
</tr>
</tbody>
</table>

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

Bit 15 PES: CPU2 program / erase suspend request
0: Flash memory program and erase operations granted.
1: Any new flash memory program and erase operation is suspended until this bit and the same bit in Flash memory access control register (FLASH_ACR) are cleared. The PESD bit in both the Flash memory status register (FLASH_SR) and Flash memory CPU2 status register (FLASH_C2SR) is set when at least one PES bit in FLASH_ACR or FLASH_C2ACR is set.

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

Bit 11 ICRST: CPU2 Instruction cache reset
0: CPU2 Instruction cache is not reset
1: CPU2 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: CPU2 Instruction cache enable
0: CPU2 Instruction cache is disabled
1: CPU2 Instruction cache is enabled

Bit 8 PRFTEN: CPU2 Prefetch enable.
0: CPU2 prefetch is disabled
1: CPU2 prefetch is enabled

Bits 7:0 Reserved, must be kept at reset value.

3.10.16 Flash memory CPU2 status register (FLASH_C2SR)

Address offset: 0x060
Reset value: 0x0000
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></td>
<td></td>
<td></td>
<td></td>
<td></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 31:20 Reserved, must be kept at reset value.

Bit 19 PESD: Programming / erase operation suspended.
This bit is set and reset by hardware.
Set when at least one PES bit in either Flash memory access control register (FLASH_ACR) or Flash memory CPU2 access control register (FLASH_C2ACR) is set.
Cleared when both PES bits in FLASH_ACR and FLASH_C2ACR are cleared.
When set new program or erase operations are not started.
Bit 18  **CFGBSY**: Programming or erase configuration busy

This bit is set and reset by hardware (set when first word is sent and reset when program operation completes or is interrupt by an error).
When set to ‘1’ the programming and erase settings in FLASH_PG and FLASH_PNB in *Flash memory CPU2 control register (FLASH_C2CR)* are used (busy) and cannot be changed. A flash programming or erase operation is ongoing.
When reset to ‘0’ the programming and erase settings in FLASH_PG and FLASH_PNB in *Flash memory CPU2 control register (FLASH_C2CR)* can be modified.

Bit 17  Reserved, must be kept at reset value.

Bit 16  **BSY**: Busy

This bit indicates that a flash memory operation requested by *Flash memory CPU2 control register (FLASH_C2CR)* is in progress. This is set on the beginning of a flash memory operation and reset when the operation finishes or when an error occurs.

Bit 15  Reserved, must be kept at reset value.

Bit 14  **RDERR**: PCROP read error

Set by hardware when an address to be read through the D-bus 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  **MISSERR**: Fast programming data miss error

In fast programming mode, 64 double words (512 bytes) must be sent to the flash memory successively, and the new data must be sent to the flash memory 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.

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.
3.10.17 Flash memory CPU2 control register (FLASH_C2CR)

Address offset: 0x064
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 memory CPU2 status register (FLASH_C2SR) is set.

When the PESD bit in Flash memory CPU2 status register (FLASH_C2SR) is cleared, the register write access is stalled until the CFGBSY bit is cleared (by the other CPU).

When the PESD bit in Flash memory CPU2 status register (FLASH_C2SR) is set, the register write access causes a bus error.

When the PESD bit in Flash memory CPU2 status register (FLASH_C2SR) is set, but there is no ongoing programming or erase operation, the register write access is completed. The requested program or erase operation is suspended, the BSY/CFGBSY asserted and remains 1 until suspend has been deactivated. Consequently, PESD bit goes back to 0 and the suspended operation completes.

<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>
Embedded flash memory (FLASH)  

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RDERRIE**: PCROP read error interrupt enable
This bit enables the interrupt generation when the RDERR bit in the FLASH_SR is set to 1.
0: PCROP read error interrupt disabled
1: PCROP read error interrupt enabled

Bit 25 **ERRIE**: Error interrupt enable
This bit enables the interrupt generation when the OPERR bit in the FLASH_SR is set to 1.
0: OPERR error interrupt disabled
1: OPERR error interrupt enabled

Bit 24 **EOPIE**: End of operation interrupt enable
This bit enables the interrupt generation when the EOP bit in the FLASH_SR is set to 1.
0: EOP Interrupt disabled
1: EOP Interrupt enabled

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **FSTPG**: Fast programming
0: Fast programming disabled
1: Fast programming enabled

Bit 17 Reserved, must be kept at reset value.

Bit 16 **STRT**: Start
This bit triggers an erase operation when set. If MER and PER bits are reset and the STRT bit is set, an unpredictable behavior may occur without generating any error flag. This condition must be forbidden.
This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_SR. Starting operations by the CPU1, involving secure flash memory pages is rejected and a bus error generated.

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

Bits 10:3 **PNB[7:0]**: Page number selection
These bits select the page to erase:
0x00: page 0
0x01: page 1
...
0x7F: page 127
... (STM32WB55xx only)
0xFF: page 255 (STM32WB55xx only)

Bit 2 **MER**: 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 programming disabled
1: Flash programming enabled

3.10.18 Secure flash memory start address register (FLASH_SFR)
Address offset: 0x080
This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error generated. On any read access the register value is returned.

Written values are only taken into account after OBL.

### 3.10.19 Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR)

Address offset: 0x084

Reset value: 0xXXXX XXXX

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error generated. On any read access the register value is returned.

Written values are only taken into account after OBL.
Bit 31 **C2OPT**: CPU2 boot reset vector memory selection.
   0: SBRV offset addresses SRAM1 or SRAM2, from start address 0x2000 0000. SBRV value must be kept within the SRAM area.
   1: SBRV offset addresses flash memory, from start address 0x0800 0000.

Bit 30 **NBRSD**: Non-backup SRAM2b security disable.
   NBRSD = 1: SRAM2b is non-secure
   NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bits 29:25 **SNBRSA[4:0]**: Secure non-backup SRAM2b start address
   NBRSD = 0: SRAM2b is secure. SNBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure non-backup SRAM2b area.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **BRSD**: Backup SRAM2a security disable.
   BRSD = 1: SRAM2a is non-secure
   BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 22:18 **SBRSA[4:0]**: Secure backup SRAM2a start address
   BRSD = 0: SRAM2a is secure. SBRSA[4:0] contains the start address of the first 1 Kbyte page of the secure backup SRAM2a area.

Bits 17:0 **SBRV[17:0]**: CPU2 boot reset vector
   Contains the word aligned CPU2 boot reset start address offset within the selected memory area by C2OPT.
### 3.10.20  FLASH register map

#### Table 18. Flash interface register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset (Hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>FLASH_ACR</td>
<td>0x000</td>
<td>Reset value 0 0 0 0 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</td>
</tr>
<tr>
<td>0x04</td>
<td>Reserved</td>
<td>0x004</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x08</td>
<td>FLASH_KEYR</td>
<td>0x008</td>
<td>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</td>
</tr>
<tr>
<td>0x0C</td>
<td>FLASH_OPTKEYR</td>
<td>0x00C</td>
<td>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</td>
</tr>
<tr>
<td>0x10</td>
<td>FLASH_SR</td>
<td>0x010</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x14</td>
<td>FLASH_CR</td>
<td>0x014</td>
<td>Reset value 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x18</td>
<td>FLASH_ECCR</td>
<td>0x018</td>
<td>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</td>
</tr>
<tr>
<td>0x20</td>
<td>FLASH_Optr</td>
<td>0x020</td>
<td>Reset value X X X X X X X X X X X X X X X X X X X</td>
</tr>
<tr>
<td>0x24</td>
<td>FLASH_PCR01ASR</td>
<td>0x024</td>
<td>Reset value X X X X X X X X X X X X X X X X X X X X</td>
</tr>
<tr>
<td>0x28</td>
<td>FLASH_PCR01AER</td>
<td>0x028</td>
<td>Reset value X X X X X X X X X X X X X X X X X X X X</td>
</tr>
<tr>
<td>0x2C</td>
<td>FLASH_WRP1A</td>
<td>0x02C</td>
<td>Reset value X X X X X X X X X X X X X X X X X X X X</td>
</tr>
</tbody>
</table>
### Table 18. Flash interface register map and reset values (continued)

| Offset | Register | 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 |
|--------|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0x030  | FLASH_WRP1BR | WRP1B_END[7:0] |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       |        |        |        |        |        |        |        |        |        |
| 0x034  | FLASH_PCROP1BSSR | PCROP1B_STRT[8:0] |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       |        |        |        |        |        |        |        |        |        |
| 0x038  | FLASH_PCROP1BER | PCROP1B_END[8:0] |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       |        |        |        |        |        |        |        |        |        |
| 0x03C  | FLASH_IPCDBR | IPCCDBA[13:0] |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       |        |        |        |        |        |        |        |        |        |
| 0x05C  | FLASH_C2ACR |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | 0       | 0       | 1       | 0       |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x060  | FLASH_C2SR |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       |        |        |        |        |        |        |        |        |        |        |        |
| 0x064  | FLASH_C2CR |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       | 0       |        |        |        |        |        |        |        |        |        |        |        |        |        |
| 0x080  | FLASH_SFR | DDS | FSD | SFSA[7:0] |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
|        |           | Reset value | 1       | 0       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       | X       |        |        |        |        |        |        |        |        |        |        |        |
| 0x084  | FLASH_SRRVR |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |

Refer to Section 2.2 on page 66 for the register boundary addresses.
4 Radio system

4.1 Introduction

The system is ultra low power compliant with the Bluetooth® core specification BLE5.4 and IEEE 802.15.4 standard.

The radio system consists of a 2.4 GHz RF front end and a Bluetooth® Low Energy (BLE) and IEEE 802.15.4 physical layer controller. The system is controlled from the CPU2 that contains the radio lower protocol software layers. Interface to the application running on the CPU2 is provided via mailbox message system.

4.2 Main features

- 2.4 GHz RF transceiver supporting:
  - Bluetooth® BLE5.4 (2 Mbps) standard support
  - IEEE 802.15.4-2011 standard support
- Programmable output power
- RSSI
- Integrated balun
- Bluetooth® BLE5.4 features:
  - GAP: central, peripheral, observer and broadcaster roles
  - Simultaneous multiple role support
  - Master / slave support
  - ATT / GATT: client and server
4.3 Radio system functional description

4.3.1 General description

The block diagram of the Radio system is shown in Figure 5.

**Figure 5. Radio system block diagram**

The maximum default transmit output power can be supported with a default VFBSMPS supply level. For higher output power the VFBSMPS supply level must be increased (see the device data sheet for more information).

Notes:
- UFQFN48 and VFQFN68: VSS through exposed pad, and VSSRF pin must be connected to ground plane
- WLCSP100: VSSRF pins must be connected to ground plane

The maximum default transmit output power can be supported with a default VFBSMPS supply level. For higher output power the VFBSMPS supply level must be increased (see the device data sheet for more information).
5 Cyclic redundancy check calculation unit (CRC)

5.1 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.

5.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  \[X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1\]
  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
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes
5.3 CRC functional description

5.3.1 CRC block diagram

![Figure 6. CRC calculation unit block diagram]

5.3.2 CRC internal 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>

5.3.3 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 accesses are allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written 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, 0x1A2B3C4D input data are 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, 0x11223344 output data are converted to 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.

*Note:* The type of an even polynomial is \(X+X^2+\ldots+X^n\), while the type of an odd polynomial is \(1+X+X^2+\ldots+X^n\).

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.
5.4 **CRC registers**

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

### 5.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>Data register bits</td>
</tr>
<tr>
<td></td>
<td>This register is used to write new data to the CRC calculator.</td>
</tr>
<tr>
<td></td>
<td>It holds the previous CRC calculation result when it is read.</td>
</tr>
<tr>
<td></td>
<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>

### 5.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>General-purpose 32-bit data register bits</td>
</tr>
<tr>
<td></td>
<td>These bits can be used as a temporary storage location for four bytes.</td>
</tr>
<tr>
<td></td>
<td>This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register</td>
</tr>
</tbody>
</table>
5.4.3 CRC control register (CRC_CR)

Address offset: 0x08
Reset value: 0x0000 0000

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

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
This bitfield controls 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.
5.4.4 CRC initial value (CRC_INIT)
Address offset: 0x10
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>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 CRC_INIT[31:0]: Programmable initial CRC value
This register is used to write the CRC initial value.

5.4.5 CRC polynomial (CRC_POL)
Address offset: 0x14
Reset value: 0x04C1 1DB7

<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 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.
### 5.4.6 CRC register map

**Table 20. CRC register map and reset values**

| Offset | Register name | 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   | CRC_DR       |                      | 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  |
|        |              |                      | 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 1 1 1 |
| 0x04   | CRC_IDR      |                      | 0  | 0  | 0  | 0  | 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 |
| 0x08   | CRC_CR       |                      | 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  |
|        |              |                      | 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   | CRC_INIT     |                      | 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  |
|        |              |                      | 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   | CRC_POL      |                      | 0  | 0  | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 1  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 1  | 1  |
|        |              |                      | Reset value | 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 |

Refer to *Section 2.2 on page 66* for the register boundary addresses.
6 Power control (PWR)

6.1 Power supplies

The devices require a $V_{DD}$ operating voltage supply between 1.71 V and 3.6 V. Several independent supplies ($V_{DDSMPS}$, $V_{FBFSMPS}$, $V_{DDA}$, $V_{DDRF}$, $V_{DDUSB}$, $V_{LCD}$) can be provided for specific peripherals:

- $V_{DD} = 1.71$ V to 3.6 V
  - $V_{DD}$ is the external power supply for the I/Os, the system analog blocks such as reset, power management, internal clocks and low power regulator. It is provided externally through VDD pins.

- $V_{DDSMPS} = 1.71$ V to 3.6 V
  - $V_{DDSMPS}$ is the external power supply for the SMPS step-down converter. It is provided externally through VDDSMPS supply pin, and must be connected to the same supply as $V_{DD}$.

- $V_{FBFSMPS} = 1.40$ V to 3.6 V
  - $V_{FBFSMPS}$ is the external power supply for the main and RF system regulators. It is provided externally through VFBSMPS pin, and may be supplied through either the SMPS step-down converter or connected to the same supply as $V_{DD}$.

- $V_{DDA} = 1.62$ V (ADCs/COMPs) to 2.4 V (VREFBUF) to 3.6 V
  - $V_{DDA}$ is the external analog power supply for A/D converters, D/A converters, voltage reference buffer, operational amplifiers and comparators. The $V_{DDA}$ voltage level is independent from the $V_{DD}$ voltage and must preferably be connected to $V_{DD}$ when these peripherals are not used. Note that VREFBUF is available only on STM32WB55xx devices.

- $V_{DDRF} = 1.71$ V to 3.6 V
  - $V_{DDRF}$ is the external power supply for the Radio. It is provided externally through the VDDRF pin, and must be connected to the same supply as $V_{DD}$.

- $V_{DDUSB} = 3.0$ V to 3.6 V
  - $V_{DDUSB}$ is the external independent power supply for USB transceivers. The $V_{DDUSB}$ voltage level is independent from the $V_{DD}$ voltage and must preferably be connected to $V_{DD}$ when the USB is not used.

- $V_{LCD} = 2.5$ V to 3.6 V (available only on STM32WB55xx devices)
  - The LCD controller can be powered either externally through the VLCD pin, or internally from an internal voltage generated by the embedded step-up converter, which can generate a voltage up to 3.6 V if $V_{DD}$ is higher than 2.0 V. VLCD is multiplexed with PC3, which can be used as GPIO when the LCD is not used.

- $V_{BAT} = 1.55$ V to 3.6 V
  - $V_{BAT}$ is the power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when $V_{DD}$ is not present.
VREF-, VREF+ (available only on STM32WB55xx devices)

VREF+ is the input reference voltage for the ADC. It is also the output of the internal voltage reference buffer when enabled.

- \( V_{DDA} < 2 \text{ V} \): VREF+ must be equal to \( V_{DDA} \)
- \( V_{DDA} \geq 2 \text{ V} \): VREF+ must be between 2 V and \( V_{DDA} \)

VREF+ can be grounded when ADC is not active.

The internal voltage reference buffer supports two output voltages, configured with VRS bit in the VREFBUF_CSR register:

- \( V_{REF+} \approx 2.048 \text{ V} \): this requires \( V_{DDA} \) equal to or higher than 2.4 V
- \( V_{REF+} \approx 2.5 \text{ V} \): this requires \( V_{DDA} \) equal to or higher than 2.8 V

VREF+ pin is not available on all packages, when not available it is internally bonded to VDDA. When the VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled (refer to the datasheet for pinout descriptions).

During power up and power down, the following power sequence is required:

When \( V_{DD} < 1 \text{ V} \) the other power supplies (\( V_{DDA}, V_{DDUSB} \) and \( V_{LCD} \)) must remain below \( V_{DD} + 0.3 \text{ V} \). During the power down \( V_{DD} \) can temporarily become lower than the other supplies only if the energy provided to the MCU remains below 1 mJ. This allows the external decoupling capacitors to discharge with different time constants.

When \( V_{DD} \geq 1 \text{ V} \) all power supplies become independent.

An embedded linear voltage regulator is used to supply the internal digital power \( V_{CORE} \). \( V_{CORE} \) is the power supply for digital peripherals, SRAM1 and SRAM2. The flash memory is supplied by \( V_{CORE} \) and \( V_{DD} \).
As shown in **Figure 8**, different supply configurations are supported, controlled according to the scheme of **Table 21**.

**Figure 8. Supply configurations**

The SMPS step-down converter operating mode depends upon the SMPSEN setting in **PWR control register 5 (PWR_CR5)** and upon the system operating modes Run, Stop0, Stop1, Stop2, Standby, and Shutdown.
Two specific features have been added to this step-down SMPS in association with the supported low power modes:

- **BYPASS mode**: capability for the SMPS to be bypassed (the current continues to flow through its coil). The \( V_{DD_{SMPS}} \) voltage is directly connected to \( V_{LX_{SMPS}} \). An additional hardware mechanism allows the SMPS to automatically switch to BYPASS mode if \( V_{DD} \) falls below a given level \( V_{BORH} \), ensuring smooth operation even at very low operating voltages.

- **OPEN mode**: in this mode the SMPS output \( (V_{LX_{SMPS}}) \) is open from any circuitry and the load of the SMPS bulk capacitance is kept. This makes it possible to speed up the wake-up time, by having the bulk capacitance precharged.

The internal state machine manages these states automatically when entering and exiting the low power modes.

After a POR reset the SMPS step-down converter is in Bypass mode.

During Stop1, Stop2, and Standby the SMPS step-down converter is put in the Open mode by hardware (see Table 21). When exiting from low-power modes (except Shutdown) the SMPS step-down converter is set by hardware to the mode selected by the SMPSEN bit (SMPS mode or Bypass). The SMPSEN bit in **PWR control register 5 (PWR_CR5)** is retained in Standby mode.

When the SMPS step-down converter is disabled SMPSEN is 0, the SMPS step-down converter is in Bypass mode and the application LDO mode can be used.

<table>
<thead>
<tr>
<th>Supply configuration</th>
<th>SMPSEN</th>
<th>System Operating mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>0</td>
<td>NA</td>
<td>SMPS in Bypass mode</td>
</tr>
<tr>
<td>SMPS supply</td>
<td>1(^{(1)})</td>
<td>Run</td>
<td>SMPS in SMPS mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop1</td>
<td>SMPS in Open mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stop2</td>
<td>Standby and Shutdown</td>
</tr>
<tr>
<td>LDO supply</td>
<td>0(^{(2)})</td>
<td>Any</td>
<td>SMPS in Bypass mode</td>
</tr>
</tbody>
</table>

1. When SMPS step-down converter SMPS mode is enabled, it is good practice to enable the BORH to monitor the supply.
2. In LDO supply the SMPS must not be enabled, or the product can be permanently damaged.

Table 21. Supply configuration control

When SMPS step-down converter SMPS mode is enabled, it is good practice to enable the BORH to monitor the supply.
6.1.1 Independent analog peripherals supply

To improve ADC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB.

The analog peripherals voltage supply input is available on a separate VDDA pin. An isolated supply ground connection is provided on VSSA pin.

The VDDA supply voltage can be different from VDD. The presence of VDDA must be checked before enabling any of the analog peripherals supplied by VDDA (A/D converter, comparators, voltage reference buffer).

The VDDA supply can be monitored by the Peripheral voltage monitoring, and compared with one threshold (1.65 V for PVM3), refer to Section 6.2.3 for more details.

When a single supply is used, VDDA can be externally connected to VDD through the external filtering circuit in order to ensure a noise-free VDDA reference voltage.

The noise performance of analog signals may be impacted by the integrated SMPS switching. To prevent this the SMPS may be switched off on the fly when measuring analog signals.

ADC reference voltage (available only on STM32WB55xx)

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to VREF+, a separate reference voltage lower than VDDA. VREF+ is the highest voltage, represented by the full scale value, for an analog input (ADC) signal.

VREF+ can be provided either by an external reference or by an internal buffered voltage reference (VREFBUF).

The internal voltage reference is enabled by setting the ENVR bit in the VREFBUF control and status register (VREFBUF_CSR). The 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 voltage reference can also provide the voltage to external components through VREF+ pin. Refer to the device datasheet and to Section 17: Voltage reference buffer (VREFBUF) for further information.

6.1.2 Independent USB transceivers supply

The USB transceivers are supplied from a separate VDDUSB power supply pin. VDDUSB range is from 3.0 V to 3.6 V and is completely independent from VDD or VDDA.

After reset, the USB features supplied by VDDUSB are logically and electrically isolated and therefore are not available. The isolation must be removed before using the USB FS peripheral, by setting the USV bit in the PWR control register 2 (PWR_CR2), once the VDDUSB supply is present.

The VDDUSB supply is monitored by the Peripheral voltage monitoring (PVM1) and compared with the internal reference voltage (VREFINT ~ 1.2 V), refer to Section 6.2.3 for more details.

6.1.3 Independent LCD supply (available only on STM32WB55xx)

The VLCD pin is provided to control the contrast of the glass LCD. This pin can be used in two ways:

- it can receive from an external circuitry the desired maximum voltage that is provided
on segment and common lines to the glass LCD by the microcontroller
it can also be used to connect an external capacitor that is used by the microcontroller
for its voltage step-up converter. This step-up converter is controlled by software to
provide the desired voltage to segment and common lines of the glass LCD.

The voltage provided to segment and common lines defines the contrast of the glass LCD
pixels. This contrast can be reduced when the user configures the dead time between
frames.

When an external power supply is provided to the VLCD pin, it must be in the range
from 2.5 V to 3.6 V. It does not depend on VDD.

When the LCD is based on the internal step-up converter, the VLCD pin should be
connected to a capacitor (see the product datasheet for further information).

6.1.4 Battery backup domain

To retain the content of the Backup registers and supply the RTC function when VDD 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 unit, the LSE oscillator and the PC13 to PC15 I/Os, allowing
the RTC to operate even when the main power supply is turned off. The switch to the VBAT
supply is controlled by the power-down reset embedded in the Reset block.

---

Warning: During tRSTTEMPO (temporization at VDD startup) or after a PDR
has been detected, the power switch between VBAT and VDD
remains connected to VBAT.
During the startup phase, if VDD is established in less than
tRSTTEMPO (refer to the datasheet for its value) and
VDD > VBAT + 0.6 V, a current may be injected into VBAT
through an internal diode connected between VDD and the
power switch (VBAT).
If the power supply / battery connected to the VBAT pin
cannot support this current injection, it is recommended to
connect an external low-drop diode between this power
supply and the VBAT pin.

---

If no external battery is used in the application, it is recommended to connect VBAT to VDD
supply, and add a 100 nF external ceramic decoupling capacitor on VBAT pin.

When the backup domain is supplied by VDD (analog switch connected to VDD), the
following pins are available:
- PC13 (STM32WB55xx only), PC14 and PC15, which can be used as GPIO pins
- PC13 (STM32WB55xx only), PC14 and PC15, which can be configured by RTC or LSE
  (refer to Section 29.4: RTC functional description)
- PA0/RTC_TAMP2 and PC12/RTC_TAMP3 (STM32WB55xx only) when configured by the
  RTC as tamper pins

Note: As the analog 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 must be limited to 2 MHz with a
maximum load of 30 pF, and these I/Os cannot be used as current sources (e.g. to drive a LED).

When the backup domain is supplied by VBAT (analog switch connected to VBAT because VDD is not present), the following functions are available:
- PC13 (STM32WB55xx only), PC14 and PC15 can be controlled only by RTC or LSE (refer to Section 29.4: RTC functional description)
- PA0/RTC_TAMP2 and PC12/RTC_TAMP3 (STM32WB55xx only) when configured by the RTC as tamper pins

Backup domain access

After a system reset, the backup domain (RTC registers and backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, set the DBP bit in the PWR control register 1 (PWR_CR1) to enable access to the backup domain.

VBAT battery charging

When VDD 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 kΩ resistor or through a 1.5 kΩ resistor, depending upon the VBRS bit value in the PWR control register 4 (PWR_CR4).

The battery charging is enabled by setting VBE bit in the PWR control register 4 (PWR_CR4), and automatically disabled in VBAT mode.

6.1.5 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the backup domain. The main regulator output voltage (VCORE) can be programmed by software to two different power ranges (Range 1 and Range 2) to optimize the consumption depending on the system maximum operating frequency (refer to Section 8.2.10: Clock source frequency versus voltage scaling and to Section 3.3.4: Read access latency).

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

- In Run, Sleep and Stop0 modes, both regulators are enabled and the main regulator (MR) supplies full power to the VCORE 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 VCORE domain, preserving the contents of the registers and internal SRAM1 and SRAM2.
- In Stop1 and Stop2 modes the main regulator is off and the low-power regulator (LPR) supplies low power to the VCORE domain, preserving the contents of the registers and of internal SRAM1 and SRAM2.
- In Standby mode with SRAM2a content preserved (RRS bit is set in the PWR control register 3 (PWR_CR3)), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM2a only. The core and digital peripherals (except Standby circuitry and backup domain), SRAM1 and SRAM2b are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers and of
SRAM1 and SRAM2 is lost except for the Standby circuitry and the backup domain.

In Shutdown mode, both regulators are powered off. When exiting from Shutdown mode, a power-on reset is generated. Consequently, the contents of the registers and of both SRAM1 and SRAM2 is lost, except for the backup domain.

### 6.1.6 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique that 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 is used to improve the device performance.

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

- **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 memory 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 possible.

Voltage scaling is selected through the VOS bit in the PWR control register 1 (PWR\_CR1).

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

1. Reduce the system frequency to a value lower or equal to 16 MHz.
2. Adjust number of wait states according new frequency target in Range2 (LATENCY bits in the FLASH\_ACR).
3. Select Range 2 in the VOS bits in the **PWR control register 1 (PWR\_CR1)**.

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

1. Select Range 1 in the VOS bits in the **PWR control register 1 (PWR\_CR1)**.
2. Wait until the VOSF flag is cleared in the **PWR status register 2 (PWR\_SR2)**.
3. Adjust number of wait states according new frequency target in Range1 (LATENCY bits in the FLASH\_ACR).
4. Increase the system frequency.

### 6.2 Power supply supervisor

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

The device has an integrated power-on reset / power-down reset, coupled with a brown-out reset circuitry.

Five BOR thresholds can be selected through option bytes.
The BOR can be used in two operating modes, configured by the BORHC bit in PWR control register 5 (PWR_CR5):

- **Reset mode**, where a system reset is generated when the \( V_{\text{DD}} \) supply is below the selected BOR threshold.
- **Force SMPS step-down converter in Bypass mode** when the \( V_{\text{DD}} \) supply is below the selected BOR threshold. When the \( V_{\text{DD}} \) supply is below the lowest BOR level a system reset is always generated.

The BOR is active in all power modes except Shutdown mode, and cannot be disabled. The BOR mechanism needs to be enabled, and can be disabled at any time if needed.

**Reset mode**

During power-on, the BOR keeps the device under reset until the supply voltage \( V_{\text{DD}} \) reaches the specified \( V_{\text{BORx}} \) threshold. When \( V_{\text{DD}} \) drops below the selected threshold, a device reset is generated. When \( V_{\text{DD}} \) is above the \( V_{\text{BORx}} \) upper limit, the device reset is released and the system can start.

**Force SMPS step-down converter Bypass mode**

During power-on, the BOR keeps the device under reset until the supply voltage \( V_{\text{DD}} \) reaches the specified \( V_{\text{BORX}} \) threshold. When \( V_{\text{DD}} \) drops below \( V_{\text{BOR0}} \), a device reset is generated. When \( V_{\text{DD}} \) is above the \( V_{\text{BORX}} \) upper limit, the device reset is released and the system can start.

To enter and during the SMPS step-down converter in SMPS mode the selected BORx (\( x = 1, 2, 3, 4 \)) threshold is used to determine if SMPS mode is used or if the SMPS step-down converter is forced in Bypass mode. When \( V_{\text{DD}} \) drops below the selected threshold, the SMPS step-down converter when entering or in SMPS mode, is forced in Bypass mode. An SMPS step-down converter force Bypass interrupt (SMPSFBF) may be generated when enabled. When \( V_{\text{DD}} \) rises above the selected BOR upper limit, a BOR interrupt (BORHF) may be generated when enabled. It is up to software to set the SMPS step-down converter in SMPS mode.

For more details on the brown-out reset thresholds refer to the electrical characteristics section in the datasheet.
6.2.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the V\textsubscript{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the PWR control register 2 (PWR\_CR2).

The PVD can also be used to monitor a voltage level on the PVD\_IN pin. In this case the voltage level on PVD\_IN is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the PWR status register 2 (PWR\_SR2), to indicate if V\textsubscript{DD} or the voltage level on PVD\_IN 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\textsubscript{DD} or the voltage level on PVD\_IN drops below the PVD threshold, and/or when V\textsubscript{DD} or the voltage level on PVD\_IN rises above the PVD threshold, depending on EXTI Line16 rising/falling edge configuration. If the EXTI Line16 is configured to rising/falling edge sensitivity (rising/falling edge of PVDO), the interrupt is generated when V\textsubscript{DD} or the voltage level on PVD\_IN drops below/rises above the PVD threshold. As an example, the service routine can perform emergency shutdown tasks.
6.2.3 Peripheral voltage monitoring (PVM)

Only VDD is monitored by default, as it is the only supply required for all system-related functions. The other supplies (VDDA and VDDUSB) can be independent from VDD and can be monitored by the peripheral voltage monitoring (PVM).

Each PVM is a comparator between a fixed threshold VPVMx and the selected power supply. PVMOx flags indicate if the independent power supply is higher or lower than the PVMx threshold: PVMOx flag is cleared when the supply voltage is above the PVMx threshold, and is set when the supply voltage is below the PVMx threshold.

Each PVM output is connected to an EXTI line and can generate an interrupt if enabled through the EXTI registers. The PVMx output interrupt is generated when the independent power supply drops below the PVMx threshold and/or when it rises above the PVMx threshold, depending on EXTI line rising/falling edge configuration.

Each PVM can remain active in Stop0, Stop1 and Stop2 modes, and the PVM interrupt can wake up from the Stop mode.

### Table 22. PVM features

<table>
<thead>
<tr>
<th>PVM</th>
<th>Power supply</th>
<th>PVM threshold</th>
<th>EXTI line</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVM1</td>
<td>VDDUSB</td>
<td>V_PVM1 (~ 1.2 V)</td>
<td>31</td>
</tr>
<tr>
<td>PVM2</td>
<td>Not used</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PVM3</td>
<td>VDDA</td>
<td>V_PVM3 (~ 1.65 V)</td>
<td>33</td>
</tr>
<tr>
<td>PVM4</td>
<td>Not used</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
The independent supplies (VDDA and VDDUSB) are not considered as present by default, and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated sources.

- If these supplies are shorted externally to VDD, the application must assume that they are available without enabling any PVM.
- If these supplies are independent from VDD, the PVM can be enabled to confirm whether the supply is present or not.

The following sequence must be applied before using the USB_FS peripheral:

1. If VDDUSB is independent from VDD:
   a) Enable the PVM1 by setting PVME1 bit in the PWR control register 2 (PWR_CR2)
   b) Wait for the PVM1 wake-up time
   c) Wait until PVMO1 bit is cleared in the PWR status register 2 (PWR_SR2)
   d) Optional: Disable the PVM1 for consumption saving
2. Set the USV bit in the PWR control register 2 (PWR_CR2) to remove the VDDUSB power isolation.

The following sequence must be applied before using analog to digital converters, digital to analog converters, comparators, operational amplifiers, voltage reference buffer:

1. If VDDA is independent from VDD:
   a) Enable the PVM3 by setting PVME3 bit in the PWR control register 2 (PWR_CR2)
   b) Wait for the PVM3 wake-up time
   c) Wait until PVMO3 bit is cleared in the PWR status register 2 (PWR_SR2)
   d) Optional: disable the PVM3 for consumption saving
2. Enable the analog peripheral, which automatically removes the VDDA isolation.

### 6.3 CPU2 boot

Booting of the CPU2 is controlled by the C2BOOT bit in PWR control register 4 (PWR_CR4) register. This allows the CPU1 to initialize the system after a reset or wake-up from system Low-power mode, before booting the CPU2.

Following reset the CPU2 is prevented from booting by the C2BOOT bit. The CPU2 boots only once the CPU1 has set the C2BOOT bit.

When exiting from system low power modes STOP0, STOP1, STOP2 or STANDBY the C2BOOT bit controls the CPU2 booting:

- When C2BOOT = 1, after a system low power mode the CPU2 boots when it is woken up via a wake-up source.
- When C2BOOT = 0, after a system low power mode the CPU2 is prevented from booting. Instead on a CPU2 wake-up source the CPU1 is woken up via a C2HF. It is up to the CPU1 to boot the CPU2 by setting the C2BOOT bit.
When the CPU2 is prevented from booting, the wake-up from Low-power mode boot procedure is the following:

Before the CPU1 enters CSTOP mode it clears the C2BOOT bit.

When CPU1 exits CSTOP:
- When the system remains in Run mode, it sets the C2BOOT bit, and subsequently processes the wake-up event.
- When the system exits from a low power mode from a CPU1 wake-up source, it initialize the system and sets the C2BOOT bit, and subsequently processes the wake-up event.
- When the system exits from a low power mode from the C2HF wake-up source, it initializes the system and set the C2BOOT bit, and subsequently goes back to CSTOP.
There is no special wake-up procedure for the CPU2. When the CPU2 wakes up, the system has been initialized by the CPU1. So the CPU2 can directly process the wake-up event.

When the system remains in Run mode (due to the Radio system) the CPU2 wakes up from CSTOP mode independently from the C2BOOT setting.

6.4 Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset and at least one CPU is in CRun mode executing code. Low-power modes are available to save power when the CPU does not need to be kept running, for example when it is waiting for an external event. The user has to select the mode that gives the best compromise between consumption, startup time and available wake-up sources.

The individual CPUs feature two low power modes, entered by the CPU when executing WFI, WFE or on return from an exception handler when SLEEPONEXIT is enabled.

- CSleep mode: when the CPU enters low power mode and SLEEPDEEP is disabled, Arm® “sleep mode”.
- CStop mode: when the CPU enters low power mode and SLEEPDEEP is enabled, Arm® “sleepdeep mode”.

The device features several low-power modes:

- **Sleep mode**: CPU clock off, all peripherals including CPU core peripherals (among them NVIC, SysTick) can run and wake up the CPU when an interrupt or an event occurs.
- **Low-power run mode (LP run)**: This mode is achieved when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or from the flash memory. The regulator is in low-power mode to minimize the operating current.
- **Low-power sleep mode (LP sleep)**: This mode is entered from the Low-power run mode: CPU is off.
- **Stop0 mode, Stop1 mode and Stop2 mode**: the content of SRAM1, SRAM2 and of all registers is retained. All clocks in the V_CORE domain are stopped, the PLL, the MSI, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running.
  The RTC can remain active (Stop mode with RTC, Stop mode without RTC).
  Some peripherals with the wake-up capability can enable the HSI16 RC during Stop mode to detect their wake-up condition.

  In Stop2 mode, most of the V_CORE domain is put in a lower leakage mode. Stop1 offers the largest number of active peripherals and wake-up sources, a smaller wake-up time but a higher consumption compared with Stop2. In Stop0 mode, the main regulator remains ON, resulting in the fastest wake-up time, but with much higher consumption. The active peripherals and wake-up sources are the same as in Stop1 mode.

  The system clock, when exiting from Stop0, Stop1 or Stop2 mode, can be either MSI up to 48 MHz or HSI16, depending on the software configuration.

- **Standby mode**: V_CORE domain is powered off. However, it is possible to preserve the SRAM2a contents:
  - Standby mode with SRAM2a retention when bit RRS is set in the *PWR control register 3 (PWR_CR3)*. In this case, SRAM2a is supplied by the low-power regulator.
  - Standby mode when bit RRS is cleared in the *PWR control register 3 (PWR_CR3)*.
In this case the main regulator and the low-power regulator are powered off. All clocks in the V\textsubscript{CORE} domain are stopped, the PLL, the MSI, the HSI16 and the HSE are disabled. The LSI and the LSE 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 HSI16.

**Shutdown mode**: V\textsubscript{CORE} domain is powered off. All clocks in the V\textsubscript{CORE} domain are stopped, the PLL, the MSI, the HSI16, the LSI and the HSE are disabled. The LSE can be kept running. The system clock, when exiting Shutdown mode, is MSI at 4 MHz. In this mode, the supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop.

**Note:** *Stop, Standby, and Shutdown modes are only entered when both CPUs are in CStop mode.*

In addition, the power consumption in Run mode can be reduced by slowing down the system clocks, and/or by gating the clocks to the APB and AHB peripherals when they are unused.

The system operation mode depend on the CPU1, the CPU2 and the Radio sub-system operating mode. The system enters a low power mode only when all three sub-systems allow it to do so.

After a system reset the CPU1 is in CRUN mode. The CPU2 boots only if enabled by the CPU1 via the C2BOOT register bit. As long as the CPU1 does not boot the CPU2, the device operates as a single CPU system. The CPU1 can enter and wake up from system low power modes on its own.

When CPU2 has boot, the CPU1, CPU2 and Radio sub-systems can enter and wake up from system low power modes on their own. The different wake-up sources for the different sub-systems are detailed in Table 23.

The system low power mode to enter depends on the allowed mode selected by both CPUs in the LPMS bits of **PWR control register 1 (PWR\_CR1)** and **PWR CPU2 control register 1 (PWR\_C2CR1)**. This is also valid when CPU2 is kept in hold by C2BOOT.

**Figure 12** shows the operating modes state diagram. The CPU1, CPU2 and Radio sub-systems operate interdependently, according to their own sub-system states. Each sub-system has its own wake-up sources, to wake up from Stop and Standby modes. For the device to be in Stop, Standby or Shutdown mode, all three sub-systems need to be in CStop. When one sub-system enters CRun mode, the device enters Run mode.

### Table 23. Sub-system low power wake-up sources

<table>
<thead>
<tr>
<th>Wake-up source</th>
<th>CPU1</th>
<th>CPU2</th>
<th>Radio</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTI</td>
<td>From Stop modes</td>
<td>From Stop modes</td>
<td>Not available</td>
</tr>
<tr>
<td>RTC</td>
<td>From Stop and Standby modes</td>
<td>From Stop and Standby modes</td>
<td>Not available</td>
</tr>
<tr>
<td>WKUP</td>
<td>From Stop and Standby modes</td>
<td>From Stop and Standby modes</td>
<td>Not available</td>
</tr>
<tr>
<td>SMPS</td>
<td>From Stop</td>
<td>From Stop</td>
<td>Not available</td>
</tr>
<tr>
<td>RADIO</td>
<td>From Stop</td>
<td>From Stop</td>
<td>Not available</td>
</tr>
<tr>
<td>RF\textsubscript{WAKEUP}</td>
<td>Not available</td>
<td>From Stop and Standby modes</td>
<td>From Stop and Standby modes</td>
</tr>
</tbody>
</table>
Figure 12. Low-power modes possible transitions

- **RUN**
  - Wakeup from STOP with CPU HOLD
  - C1_wakeup
  - CPU1 CRUN or CSLEEP
  - CPU2 CRUN or CSLEEP
  - Radio CRUN or CSTOP

- **LP-RUN**
  - CPU2 sub-system having allocated peripheral in the HCLK1 domain

- **STOP**
  - C1STOP
  - CPU1 CSTOP
  - CPU2 CRUN or CSLEEP
  - Radio CRUN or CSTOP

- **STOP0**
  - CPU1 CRUN or CSLEEP
  - CPU2 CSTOP
  - Radio CSTOP

- **STOP1**
  - HCLK1 RUN
  - HCLK4 RUN

- **STOP2**
  - C2_wakeup
  - C1_wakeup

- **LP-STOP**
  - HCLK1 peripheral allocation
  - C1STOP
  - C2STOP

- **STANDBY**
  - Wakeup from BLE-WKUP
  - Wakeup CPU from STM32-WKUP

- **SHUTDOWN**
  - Wakeup CPU from STM32-WKUP
  - Wakeup from BLE-WKUP

### Table 24. 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 regulator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sleep (Sleep-now or Sleep-on-exit)</strong></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 ON, LPR ON</td>
</tr>
<tr>
<td></td>
<td>WFE</td>
<td>Wakeup event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Low-power run</strong></td>
<td>Set LPR bit</td>
<td>Clear LPR bit</td>
<td>Same as Low-power run clock</td>
<td>None</td>
<td>MR OFF, LPR ON</td>
</tr>
<tr>
<td><strong>Low-power sleep</strong></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, LPR OFF</td>
</tr>
<tr>
<td></td>
<td>Set LPR bit + WFE</td>
<td>Wakeup event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Stop0</strong></td>
<td>LPMS = &quot;000&quot; + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>Any EXTI line (configured in the EXTI registers). Specific peripherals events</td>
<td>HSI16 when STOPWUCK = 1 in RCC_CFGGR. MSI with the frequency before entering Stop mode when STOPWUCK = 0.</td>
<td>All clocks OFF except LSI and LSE</td>
<td>MR ON, LPR OFF</td>
</tr>
<tr>
<td><strong>Stop1</strong></td>
<td>LPMS = &quot;001&quot; + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td></td>
<td></td>
<td></td>
<td>MR OFF, LPR OFF</td>
</tr>
<tr>
<td><strong>Stop2</strong></td>
<td>LPMS = &quot;010&quot; + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td></td>
<td></td>
<td></td>
<td>MR OFF, LPR OFF</td>
</tr>
<tr>
<td><strong>Standby with SRAM2a</strong></td>
<td>LPMS = &quot;011&quot; + Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>WKUP pin edge, RTC event, LSECSS, external reset in NRST pin, IWDG reset</td>
<td>HSI16</td>
<td>All clocks OFF except LSI and LSE</td>
<td>MR OFF, LPR OFF</td>
</tr>
<tr>
<td><strong>Standby</strong></td>
<td>LPMS = &quot;011&quot; + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>WKUP pin edge, RTC event, LSECSS, external reset in NRST pin, IWDG reset</td>
<td>HSI16</td>
<td>All clocks OFF except LSI and LSE</td>
<td>Mr OFF, LPR OFF</td>
</tr>
<tr>
<td><strong>Shutdown</strong></td>
<td>LPMS = &quot;1--&quot; + SLEEPDEEP bit + WFI or Return from ISR or WFE</td>
<td>WKUP pin edge, RTC event, external reset in NRST pin</td>
<td>MSI 4 MHz</td>
<td>All clocks OFF except LSE</td>
<td>MR OFF, LPR OFF</td>
</tr>
</tbody>
</table>

1. Refer to Table 25.
Table 25. Functionalities depending on system operating mode

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Run</th>
<th>Sleep</th>
<th>Low-power run</th>
<th>Low-power sleep</th>
<th>Stop0</th>
<th>Stop1</th>
<th>Stop2</th>
<th>Standby</th>
<th>Shutdow</th>
<th>VBAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CPU2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Radio-system (BLE, 802)</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y^2</td>
<td>-</td>
</tr>
<tr>
<td>Flash memory</td>
<td>Y</td>
<td>Y</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>SRAM1</td>
<td>Y</td>
<td>O^4</td>
<td>Y</td>
<td>O^4</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>O</td>
</tr>
<tr>
<td>SRAM2a</td>
<td>Y</td>
<td>O^4</td>
<td>Y</td>
<td>O^4</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>O^3</td>
</tr>
<tr>
<td>SRAM2b</td>
<td>Y</td>
<td>O^4</td>
<td>Y</td>
<td>O^4</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>O(2)</td>
</tr>
<tr>
<td>QUADSPI</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Backup registers</td>
<td>Y</td>
<td>Y</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>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>Y</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>Brown-out SMPS force bypass (BOR)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</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>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>Peripheral voltage monitor</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>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>DMAx (x=1, 2)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>High speed internal (HSI16)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(5)</td>
<td>O(5)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Oscillator HSI48</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</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>
<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>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>
<td>O</td>
<td>O(2)</td>
</tr>
<tr>
<td>Multi-speed internal (MSI)</td>
<td>O</td>
<td>O</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</td>
<td>O</td>
<td>-</td>
<td>-</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>O</td>
<td>O</td>
<td>-</td>
</tr>
<tr>
<td>RTC / Auto wake-up</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>
<td>O</td>
<td>O(3)</td>
</tr>
<tr>
<td>Number of RTC tamper pins</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>LCD^6</td>
<td>O(7)</td>
<td>O(7)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>USB FS</td>
<td>O(7)</td>
<td>O(7)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>USART1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
</tr>
<tr>
<td>Low-power UART (LPUART)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
<td>O(8)</td>
</tr>
</tbody>
</table>

Note: Y = Yes, - = No, O = Optional, 0 = OFF, R = Resistor, VBAT = Voltage at Battery

^1 Table 25. Functionalities depending on system operating mode

^2 Y^2 indicates that the functionality is active in Stop2 mode as well as in Standby and Shutdow modes.
### Table 25. Functionalities depending on system operating 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>Stop0</th>
<th>Stop1</th>
<th>Stop2</th>
<th>Standby</th>
<th>Shutdown</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Wakeup capability</td>
<td>Wakeup capability</td>
<td>Wakeup capability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2C1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>O(^{(8)})</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I2C3</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O(^{(8)})</td>
<td>O(^{(8)})</td>
<td>O(^{(8)})</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SPI(_x) ((x=1, 2))(^{(6)})</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ADC1</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COMP(_x) ((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>-</td>
<td>-</td>
</tr>
<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>
<td>-</td>
</tr>
<tr>
<td>Timers (T(\text{IMx}))</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</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>
<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>
<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>O</td>
<td>O</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>
<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>
<td>-</td>
</tr>
<tr>
<td>Touch sensing controller (TSC)(^{(6)})</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>True random number generator (RNG)</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AES hardware accelerator</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</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>
<td>-</td>
</tr>
<tr>
<td>IPCC</td>
<td>O</td>
<td>-</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HSEM</td>
<td>O</td>
<td>-</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PKA</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</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>O</td>
<td>(10)</td>
</tr>
</tbody>
</table>

1. Legend: Y = Yes (enabled), O = Optional (disabled by default, can be enabled by software). R = Data retained.
   - = Not available. Gray cells indicate Wakeup capability.
2. SRAM\(_2\) content needs to be retained via the PWR\_CR3.RRS bit.
3. The SRAM\(_2\) content can optionally be retained when the PWR\_CR3.RRS bit is set.
4. The SRAM clock can be gated on or off.
5. Some peripherals with wake-up from Stop capability can request HSI\(_{16}\) to be enabled. In this case, HSI\(_{16}\) is woken up by the peripheral, and only feeds the peripheral that requested it. HSI\(_{16}\) is automatically put off when the peripheral does not need it anymore.
6. LCD, SPI2 and TSC are available only on STM32WB55xx devices.
7. Voltage scaling Range 1 only.
8. UART and LPUART reception is functional in Stop mode, and generates a wake-up interrupt on Start, address match or received frame event.

9. I2C address detection is functional in Stop mode, and generates a wake-up interrupt in case of address match.

10. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.

11. The I/Os with wake-up from Standby/Shutdown capability are PA0 and PA2, and (for STM32WB55xx only) PC5, PC12 and PC13.

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 application puts the MCU in Stop0, Stop1, Stop2, Standby or Shutdown mode while the debug features are used. This is because the CPU1 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 41.3.5: Debug and low power modes.

**6.4.1 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 8.4.3: RCC 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 WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

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

**6.4.2 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 must not exceed 2 MHz. The Radio sub-system cannot be used in low-power Run mode.

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 the Low-power run mode, proceed as follows:

1. Optional: Jump into the SRAM and power-down the flash memory by setting the FPDR bit in `PWR control register 1 (PWR_CR1)` and `PWR CPU2 control register 1 (PWR_C2CR1)`.
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 control register 1 (PWR_CR1)`.

Refer to Table 26 on how to enter the Low-power run mode.

Exiting Low-power run mode

To exit the Low-power run mode, proceed as follows (refer to Table 26):

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

### Table 26. Low-power run

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

6.4.3 Entering Low-power mode

Low power modes are entered by the MCU by executing WFI (Wait for Interrupt), or WFE (Wait for Event) instructions, or when the SLEEPONSEC bit in the CPU1 System control register is set on Return from ISR.

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

6.4.4 Exiting Low-power mode

From Sleep modes, and Stop modes the MCU exit Low-power mode depending on the way the mode was entered:

- If the WFI instruction or Return from ISR was used to enter the Low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the Low-power mode, the MCU exits the Low-power mode as soon as an event occurs. The wake-up event can be generated either by an NVIC IRQ interrupt, or by an event.
  - In the first case, when SEVONPEND = 0 in the CPU System control register, 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. Only NVIC interrupts with sufficient priority wake-up and interrupt the MCU.

When SEVONPEND = 1 in the CPU System control register, 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 wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with sufficient priority wake up and interrupt the MCU.

– In the second case, configuring an 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.

From Standby and Shutdown modes the MCU exits Low-power mode through an external reset (NRST pin), an IWDG reset, a rising/falling edge on one of the enabled WKUPx pins, or an RTC event (see Section 29: Real-time clock (RTC)), or a Radio event (for Standby only).

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.).

The system mode when the CPU wakes up from CStop mode can be determined from the CnSTOPF and CnSBF in PWR extended status and status clear register (PWR_EXTSCR).

<table>
<thead>
<tr>
<th>System mode</th>
<th>CPU1 C1SBF</th>
<th>CPU1 C1STOPF</th>
<th>CPU1 C2SBF</th>
<th>CPU1 C2STOPF</th>
<th>CPU2 C2SBF</th>
<th>CPU2 C2STOPF</th>
<th>CPU1 wake-up</th>
<th>CPU2 wake-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Wakeup from Run</td>
<td>Wakeup from Run</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Wakeup from STOP, but system is already in Run due to CPU2</td>
<td>Wakeup from Run</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Wakeup from Run</td>
<td>Wakeup from STOP, but system is already in Run due to CPU2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Wakeup from STANDBY, but system is already in Run due to CPU1</td>
<td>Wakeup from Run</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Wakeup from Run</td>
<td>Wakeup from STANDBY, but system is already in Run due to CPU1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Wakeup from STANDBY followed by STOP, but system is already in Run due to CPU2</td>
<td>Wakeup from Run</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Wakeup from Run</td>
<td>Wakeup from STANDBY followed by STOP, but system is already in Run due to CPU1</td>
</tr>
</tbody>
</table>
6.4.5 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

Sleep mode is entered according to *Entering Low-power mode*, when the SLEEPDEEP bit in the CPU System control register is cleared (see Table 28).

Exiting Sleep mode

The MCU exits from Sleep mode (see Table 28) as indicated in *Exiting Low-power mode*.

### Table 28. Sleep mode

<table>
<thead>
<tr>
<th>Sleep-now mode</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® System control register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode entry</th>
<th>On return from ISR while:</th>
</tr>
</thead>
<tbody>
<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® System control register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System mode</th>
<th>CPU1 wake-up</th>
<th>CPU2 wake-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>0 1 0 1</td>
<td>Wakeup from STOP (CPU2 still in CSTOP)</td>
</tr>
<tr>
<td></td>
<td>1 1 0 1</td>
<td>Wakeup from STOP after the system has been in STANDBY (CPU2 still in CSTOP)</td>
</tr>
<tr>
<td></td>
<td>0 1 1 1</td>
<td>Wakeup from STOP (CPU2 is still in CSTOP)</td>
</tr>
<tr>
<td>Standby</td>
<td>1 0 1 0</td>
<td>Wakeup from STANDBY (CPU2 still in CSTOP)</td>
</tr>
<tr>
<td>N.A.</td>
<td>Others</td>
<td>Not valid, does not occur</td>
</tr>
</tbody>
</table>
6.4.6 Low-power sleep mode (LP sleep)

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

Low-power sleep mode is entered from Low-power run mode as described in Section 6.4.3, when the SLEEPDEEP bit in the Cortex® System control register is cleared.

Refer to Table 29 for details on how to enter the Low-power sleep mode.

Exiting Low-power sleep mode

The low-power Sleep mode is exited as described in Section 6.4.4. When exiting Low-power sleep mode by issuing an interrupt or an event, the MCU is in Low-power run mode.

Refer to Table 29 for details on how to exit the Low-power sleep mode.

Table 29. Low-power sleep

<table>
<thead>
<tr>
<th>Low-power sleep-now mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Low-power sleep mode is entered from the Low-power run mode. 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® System control register.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Low-power sleep mode is entered from the Low-power run mode. 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® System control register.</td>
</tr>
</tbody>
</table>
6.4.7 **Stop0 mode**

The Stop0 mode is based on the CPU deep sleep mode combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop0 mode, all clocks in the \( V_{\text{CORE}} \) domain are stopped; the PLL, the MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with wake-up capability (I2Cx \((x=1, 3)\), USART1 and LPUART) can switch on the HSI16 to receive a frame, and switch off the HSI16 after receiving the frame if it is not a wake-up frame. In this case, the HSI16 clock is propagated only to the peripheral requesting it.

SRAM1, SRAM2 and register contents are preserved.

The BOR is always available in Stop0 mode. The consumption increases when thresholds higher than \( V_{\text{BOR0}} \) are used.

**I/O states in Stop0 mode**

In the Stop0 mode, all I/O pins keep the same state as in the Run mode.

**Entering Stop0 mode**

The Stop0 mode is entered according to Section 6.4.3, when the SLEEPDEEP bit in the Cortex System control register is set (see Table 30). Before entering the mode, the system clock must be set to HSI16 clock.

If flash memory programming is ongoing, the Stop0 mode entry is delayed until the operation is completed.

If an access to the APB domain is ongoing, the Stop0 mode entry is delayed until the APB access is finished.

In Stop0 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 30.3: IWDG functional description.

- Real-time clock (RTC): this is configured by the RTCEN bit in the RCC backup domain.

<table>
<thead>
<tr>
<th><strong>Low-power sleep-now mode</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode exit</td>
<td></td>
</tr>
<tr>
<td>If WFI or Return from ISR was used for entry:</td>
<td></td>
</tr>
<tr>
<td>Interrupt: refer to Table 54: CPU1 vector table and Table 55: CPU2 vector table</td>
<td></td>
</tr>
<tr>
<td>If WFE was used for entry and SEVONPEND = 0:</td>
<td></td>
</tr>
<tr>
<td>Wakeup event: refer to Section 13.5: Interrupt list</td>
<td></td>
</tr>
<tr>
<td>If WFE was used for entry and SEVONPEND = 1:</td>
<td></td>
</tr>
<tr>
<td>Interrupt even when disabled in NVIC: refer to Table 54: CPU1 vector table and Table 55: CPU2 vector table</td>
<td></td>
</tr>
<tr>
<td>Wakeup event: refer to Section 13.5: Interrupt list</td>
<td></td>
</tr>
<tr>
<td>After exiting the Low-power sleep mode, the MCU is in Low-power run mode.</td>
<td></td>
</tr>
</tbody>
</table>

| **Wakeup latency** | **None** |

<table>
<thead>
<tr>
<th><strong>Table 29. Low-power sleep (continued)</strong></th>
</tr>
</thead>
</table>

---

164/1530 RM0434 Rev 13
control register (RCC_BDCR).

Internal RC oscillator (LSI): this is configured by the LSIxON bit in the RCC control/status register (RCC_CSR).

External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the RCC backup domain control register (RCC_BDCR).

Some peripherals can be used in Stop0 mode and add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock, namely LCD, LPTIM1, LPTIM2, I2Cx (x=1, 3), USART1, LPUART.

The comparators as well as the PVMx (x=1, 3) and the PVD can be used in Stop0 mode. If not needed, they must be disabled by software to reduce power consumption.

The ADC, temperature sensor and VREFBUF buffer can consume power during the Stop0 mode, unless they are disabled before entering this mode.

**Exiting Stop0 mode**

The Stop0 mode is exited according to what indicated in Section 6.4.4.

Refer to Table 30 for details on how to exit Stop0 mode.

When exiting Stop0 mode by issuing an interrupt or a wake-up event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in RCC clock configuration register (RCC_CFGR). The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wake-up time is shorter when HSI16 is selected as wake-up system clock. The MSI selection enables wake-up at higher frequency, up to 48 MHz.

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop0 mode with HSI16. By keeping the internal regulator ON during Stop0 mode, the consumption is higher but the startup time is reduced.

When exiting Stop0 mode, the MCU is either in Run mode (Range 1 or Range 2 depending on VOS bit in PWR control register 1 (PWR_CR1)) or in Low-power run mode if the bit LPR is set in the same register.

<table>
<thead>
<tr>
<th>Stop0 mode</th>
<th>Description</th>
</tr>
</thead>
</table>
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:  
- SLEEPDEEP bit is set in Cortex System control register  
- No interrupt (for WFI) or event (for WFE) is pending  
- LPMS = "000" in PWR_CR1 and/or PWR_C2CR1 or higher  
| On Return from ISR while:  
- SLEEPDEEP bit is set in Cortex System control register  
- SLEEPONEXIT = 1  
- No interrupt is pending  
- LPMS = "000" in PWR_CR1 and/or PWR_C2CR1 or higher  |

*Note: To enter Stop0 mode, all EXTI line pending bits (in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop0 mode entry procedure is ignored and program execution continues.*
The Stop1 mode is the same as Stop0 mode except that the main regulator is OFF, and only the low-power regulator is ON. Stop1 mode can be entered from Run mode and from Low-power run mode. Before entering the mode, the system clock must be set to HSI16 clock.

Refer to Table 31 for details on how to enter and exit Stop1 mode.

### Table 30. Stop0 mode

<table>
<thead>
<tr>
<th>Stop0 mode</th>
<th>Description</th>
</tr>
</thead>
</table>
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:  
– SLEEPDEEP bit is set in Cortex System control register  
– No interrupt (for WFI) or event (for WFE) is pending  
– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher  
On Return from ISR while:  
– SLEEPDEEP bit is set in Cortex System control register  
– SLEEPONEXIT = 1  
– No interrupt is pending  
– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher |
| Mode exit  | If WFI 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 wake-up capability. Refer to Table 54: CPU1 vector table and Table 55: CPU2 vector table.  
If WFE was used for entry and SEVONPEND = 0:  
Any EXTI line configured in event mode. Refer to Section 13.5: Interrupt list.  
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 wake-up capability. Refer to Table 54: CPU1 vector table and Table 55: CPU2 vector table.  
Wakeup event: refer to Section 13.5: Interrupt list |
| Wakeup latency | Longest wake-up time between: MSI or HSI16 wake-up time and flash memory wake-up time from Stop0 mode. |

### 6.4.8 Stop1 mode

The Stop1 mode is the same as Stop0 mode except that the main regulator is OFF, and only the low-power regulator is ON. Stop1 mode can be entered from Run mode and from Low-power run mode. Before entering the mode, the system clock must be set to HSI16 clock.

Refer to Table 31 for details on how to enter and exit Stop1 mode.

### Table 31. Stop1 mode

<table>
<thead>
<tr>
<th>Stop1 mode</th>
<th>Description</th>
</tr>
</thead>
</table>
| Mode entry | WFI (Wait for Interrupt) or WFE (Wait for Event) while:  
– SLEEPDEEP bit is set in Cortex System control register  
– No interrupt (for WFI) or event (for WFE) is pending  
– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher  
On Return from ISR while:  
– SLEEPDEEP bit is set in Cortex System control register  
– SLEEPONEXIT = 1  
– No interrupt is pending  
– LPMS = “001” in PWR_CR1 and/or PWR_C2CR1 or higher |

**Note:** To enter Stop1 mode, all EXTI line pending bits (in EXTI pending register (EXTI_PR1), and EXTI pending register (EXTI_PR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop1 mode entry procedure is ignored and program execution continues.
6.4.9 Stop2 mode

The Stop2 mode is based on the CPU deepsleep mode combined with peripheral clock gating. In Stop2 mode, all clocks in the V CORE domain are stopped, the PLL, the MSI, the HSI16 and the HSE oscillators are disabled. Some peripherals with wake-up capability (I2C3 and LPUART) can switch on the HSI16 to receive a frame, and switch off the HSI16 after receiving the frame if it is not a wake-up frame. In this case the HSI16 clock is propagated only to the peripheral requesting it.

SRAM1, SRAM2 and register contents are preserved.

The BOR is always available in Stop2 mode. The consumption is increased when thresholds higher than VBOR0 are used.

**Note:** The comparators, LPUART and LPTIM1 outputs are forced to low speed (OSPEEDy = 00) during Stop2 mode.

**I/O states in Stop2 mode**

In the Stop2 mode, all I/O pins keep the same state as in the Run mode.

**Entering Stop2 mode**

The Stop2 mode is entered as described in Section 6.4.3, when the SLEEPDEEP bit in the Cortex System control register is set (see Table 32). Before entering in this mode, the system clock must be set to HSI16 clock.

Stop2 mode can only be entered from Run mode. It is not possible to enter Stop2 mode from the Low-power run mode.

If flash memory programming is ongoing, the Stop2 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop2 mode entry is delayed until the APB access is finished.
In Stop2 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 30.3: IWDG functional description.*

- **Real-time clock (RTC):** this is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*.

- **Internal RC oscillator (LSI):** this is configured by the LSIXON bit in the *RCC control/status register (RCC_CSR)*.

- **External 32.768 kHz oscillator (LSE):** this is configured by the LSEON bit in the *RCC backup domain control register (RCC_BDCR)*.

Some peripherals can be used in Stop2 mode and add consumption if they are enabled and clocked by LSI or LSE, or when they request the HSI16 clock, namely LCD, LPTIM1, I2C3, LPUART.

The comparators, the PVMx (x=1, 3) and the PVD can be used in Stop2 mode. If not needed, they must be disabled by software to reduce power consumption.

The ADC, temperature sensor and VREFBUF(a) buffer can consume power during Stop2 mode, unless they are disabled before entering this mode.

All the peripherals that cannot be enabled in Stop2 mode must be either disabled by clearing the Enable bit in the peripheral itself, or put under reset state by setting the corresponding bit in the

- *RCC AHB1 peripheral reset register (RCC_AHB1RSTR)*
- *RCC AHB2 peripheral reset register (RCC_AHB2RSTR)*
- *RCC AHB3 and AHB4 peripheral reset register (RCC_AHB3RSTR)*
- *RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)*
- *RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)*
- *RCC APB2 peripheral reset register (RCC_APB2RSTR)*
- *RCC APB3 peripheral reset register (RCC_APB3RSTR)*

**Exiting Stop2 mode**

The Stop2 mode is exited according to *Section 6.4.4* (see *Table 32*).

When exiting Stop2 mode by issuing an interrupt or a wake-up event, the HSI16 oscillator is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register (RCC_CFGR)*. The MSI oscillator is selected as system clock if the bit STOPWUCK is cleared. The wake-up time is shorter when HSI16 is selected as wake-up system clock. The MSI selection allows wake-up at higher frequency, up to 48 MHz.

When exiting Stop2 mode, the MCU is in Run mode (Range 1 or Range 2 depending on VOS bit in PWR_CR1).

---

a. Available only on Cat. 3 devices.
### 6.4.10 Standby mode

Standby mode makes it possible to achieve the lowest power consumption with BOR. It is based on the CPU deepsleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). The PLL, the HSI16, the MSI and the HSE oscillators are also switched off.

SRAM1 and register contents are lost except for registers in the Backup domain and standby circuitry (see Figure 7). SRAM2 content can be preserved if the bit RRS is set in the Power control register 3 (PWR_CR3). In this case the Low-power regulator is ON and provides the supply to SRAM2 only.

The BOR is always available in Standby mode. The consumption is increased when thresholds higher than VBOR0 are used.

### I/O states in Standby mode

In Standby mode, the I/Os can be configured either with a pull-up (refer to PWR_PUCRx registers (x=A,B,C,D,E,F,G,H)), or with a pull-down (refer to PWR_PDCRx registers (x=A,B,C,D,E,F,G,H)), or can be kept in analog state.
The RTC outputs on PC13 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. Five wake-up pins (WKUPx, x=1,2,...5) and the three RTC tampers are available.

**Entering Standby mode**

Standby mode is entered according to Section 6.4.3, when the SLEEPDEEP bit in the Cortex System control register is set.

Refer to Table 33 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 30.3: IWDG functional description.
- Real-time clock (RTC): this is configured by the RTCE bit in the Backup domain control register (RCC_BDCR).
- Internal RC oscillator (LSI): this is configured by the LSIxON bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

**Exiting Standby mode**

Standby mode is exited according to Section 6.4.4. The SBF status flag in the PWR control register 3 (PWR_CR3) indicates that the MCU was in Standby mode. All registers are reset after wake-up from Standby except for PWR control register 3 (PWR_CR3).

Refer to Table 33 for more details on how to exit Standby mode.

<table>
<thead>
<tr>
<th>Table 33. Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standby mode</strong></td>
</tr>
<tr>
<td>Mode entry</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Mode exit</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Wakeup latency</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
6.4.11 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 \( V_{CORE} \) domain is consequently powered off. The PLL, the HSI16, the MSI, the LSI and the HSE oscillators are also switched off.

SRAM1, SRAM2 and register contents are lost except for registers in the Backup domain. The BOR is not available in Shutdown mode. No power voltage monitoring is possible in this mode, therefore the switch to Backup domain 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 \((x=A,B,C,D,E,F,G,H)\)), or with a pull-down (refer to PWR_PDCRx registers \((x=A,B,C,D,E,F,G,H)\)), 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 wake-up pins \((WKUPx, x=1,2...5)\) and the three RTC tampers are available.

Entering Shutdown mode

The Shutdown mode is entered according to **Entering Low-power mode**, when the SLEEPDEEP bit in the Cortex System control register is set.

Refer to **Table 34** 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):** this is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*. Caution: in case of \(V_{DD} \) power-down the RTC content is lost.

- **External 32.768 kHz oscillator (LSE):** this is configured by the LSEON bit in the *RCC backup domain control register (RCC_BDCR)*.

Exiting Shutdown mode

The Shutdown mode is exited according to **Exiting Low-power mode**. A power-on reset occurs when exiting from Shutdown mode. All registers (except for the ones in the Backup domain) are reset after wake-up from Shutdown.

Refer to **Table 34** for more details on how to exit Shutdown mode.
6.4.12 Auto wake-up from Low-power mode

The RTC can be used to wake up the MCU from Low-power mode without depending on an external interrupt (Auto-wake-up mode). The RTC provides a programmable time base for waking up from Stop (0, 1 or 2) or Standby mode 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 RCC backup 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 oscillator is designed to add minimum power consumption.

To wake up from Stop mode with an RTC alarm event, it is necessary to:
- Configure the EXTI Line 17 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

To wake up from Stop mode with an RTC wake-up event, it is necessary to:
- Configure the EXTI Line 19 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

To wake up from Standby mode, there is no need to configure the EXTI Line 17.

The LCD Start of frame interrupt can also be used as a periodic wake-up from Stop (0, 1 or 2) mode. The LCD is not available in Standby mode.

The LCD clock is derived from the RTC clock selected by RTCSEL[1:0].

6.5 Real-time radio information

The PWR provides flags indicating the real time operation on the radio:

- IEEE802.15.4 radio activity end interrupt flag
- BLE radio activity end interrupt flag
- Critical radio phase flag
- Critical radio phase end interrupt flag

These flags may be used by the CPU1 to determine the radio activity.

The basic timing relation for the different flags is shown in Figure 12.

![Diagram](MS44444V1)

When enabled, the radio activity end interrupt flag 802AF indicates the end of an IEEE802.15.4 radio activity period, and is generated when the radio enters Cstop mode.

When enabled, the radio activity end interrupt flag BLEAF indicates the end of a BLE radio activity period, and is generated when the radio enters Cstop mode.

The critical radio phase flag indicates the critical real time phase of the radio, where access to the flash memory must not be blocked by erase or program operations (all erase and program operations are suspended by the CPU2 in the flash memory interface). The end of the critical radio phase may be triggered by the CPU2 clearing the critical radio phase flag CRPF, and is eventually cleared at the end of the IEEE802.15.4 or BLE radio activity.

When enabled, the critical radio phase end interrupt flag CRPEF indicates the end of a radio critical phase, and is generated when the radio critical phase ends.
6.6 PWR registers

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

6.6.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0200. This register is reset after wake-up from Standby mode, except for bits [2:0].

| Bit 31:15 | Reserved, must be kept at reset value. |
| Bit 14 | LPR: Low-power run |
| Note: | Stop2 mode cannot be entered when LPR bit is set. Stop1 is entered instead. |
| Bit 13:11 | Reserved, must be kept at reset value. |
| Bit 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 backup 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 |
| Bit 7:6 | Reserved, must be kept at reset value. |
| Bit 5 | FPDS: Flash memory power down mode during LPSleep for CPU1 |
| This bit selects whether the flash memory is in power down mode or idle mode when both CPUs are in Sleep mode. flash memory is set in power down mode only when the system is in LPSleep mode and the PWR_C2CR1.FPDS bit from CPU2 also allows this. |
| 0: Flash memory in Idle mode when system is in LPSleep mode |
| 1: Flash memory in power down mode when system is in LPSleep mode |
6.6.2  **PWR control register 2 (PWR_C2R)**

Address offset: 0x004

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

<table>
<thead>
<tr>
<th>Bit 31:11</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 10</td>
<td><strong>USV</strong>: $V_{DDUSB}$ USB supply valid</td>
</tr>
<tr>
<td></td>
<td>This bit is used to validate the $V_{DDUSB}$ supply for electrical and logical isolation purposes. Setting this bit is mandatory to use the USB FS peripheral. If $V_{DDUSB}$ is not always present in the application, the PVM can be used to determine whether this supply is available or not.</td>
</tr>
<tr>
<td></td>
<td>0: $V_{DDUSB}$ is not present. Logical and electrical isolation is applied to ignore this supply.</td>
</tr>
<tr>
<td></td>
<td>1: $V_{DDUSB}$ is valid.</td>
</tr>
<tr>
<td>Bit 9:7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 6</td>
<td><strong>PVME3</strong>: Peripheral voltage monitoring 3 enable: $V_{DDA}$ vs. 1.62 V</td>
</tr>
<tr>
<td></td>
<td>0: PVME3 ($V_{DDA}$ monitoring vs. 1.62 V threshold) disabled</td>
</tr>
<tr>
<td></td>
<td>1: PVME3 ($V_{DDA}$ monitoring vs. 1.62 V threshold) enabled</td>
</tr>
</tbody>
</table>
6.6.3 PWR control register 3 (PWR_CR3)

Address offset: 0x008

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes.

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

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

- **Bit 15** EIWUL: Enable internal wake-up line for CPU1
  - 0: Internal wake-up line interrupt to CPU1 disabled
  - 1: Internal wake-up line interrupt to CPU1 enabled

- **Bit 14** EC2H: Enable CPU2 Hold interrupt for CPU1
  - Enable CPU2 kept in hold, due to C2BOOT, interrupt to CPU1.
  - 0: Interrupt to CPU1 disabled
  - 1: interrupt to CPU1 enabled
Bit 13 **E802A**: Enable 802.15.4 end of activity interrupt for CPU1
0: Interrupt to CPU1 disabled
1: interrupt to CPU1 enabled

Bit 12 **EBLEA**: Enable BLE end of activity interrupt for CPU1
0: Interrupt to CPU1 disabled
1: interrupt to CPU1 enabled

Bit 11 **ECRPE**: Enable critical radio phase end of activity interrupt for CPU1
0: Interrupt to CPU1 disabled
1: interrupt to CPU1 enabled

Bit 10 **APC**: Apply pull-up and pull-down configuration from CPU1
When this bit for CPU1 or the PWR_C2CR3.APC bit for CPU2 is set, the I/O pull-up and pull-down configurations defined in the PWR_PUCRx and PWR_PDCRx registers are applied. When both bits are cleared, the PWR_PUCRx and PWR_PDCRx registers are not applied to the I/Os.

Bit 9 **RRS**: SRAM2a retention in Standby mode
0: SRAM2a powered off in Standby mode (content is lost).
1: SRAM2a powered by the low-power regulator in Standby mode (content is kept).

Bit 8 **EBORHSMPSFB**: Enable BORH and SMPS step-down converter forced in Bypass interrupts for CPU1
0: Interrupts BORHF and SMPSFBF to CPU1 disabled
1: interrupts BORHF and SMPSFBF to CPU1 enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **EWUP5**: Enable wake-up pin WKUP5 for CPU1 (available on STM32WB55xx only)
When this bit is set, the external wake-up pin WKUP5 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP5 bit in the **PWR control register 4 (PWR_CR4)**. Note that this bit is reserved in STM32WB35xx devices.

Bit 3 **EWUP4**: Enable wake-up pin WKUP4 for CPU1
When this bit is set, the external wake-up pin WKUP4 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP4 bit in the **PWR control register 4 (PWR_CR4)**.

Bit 2 **EWUP3**: Enable wake-up pin WKUP3 for CPU1 (available on STM32WB55xx only)
When this bit is set, the external wake-up pin WKUP3 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP3 bit in the **PWR control register 4 (PWR_CR4)**. Note that this bit is reserved in STM32WB35xx devices.

Bit 1 **EWUP2**: Enable wake-up pin WKUP2 for CPU1 (available on STM32WB55xx only)
When this bit is set, the external wake-up pin WKUP2 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP2 bit in the **PWR control register 4 (PWR_CR4)**. Note that this bit is reserved in STM32WB35xx devices.

Bit 0 **EWUP1**: Enable wake-up pin WKUP1 for CPU1
When this bit is set, the external wake-up pin WKUP1 is enabled and triggers an interrupt and wake-up from Stop, Standby or Shutdown event when a rising or a falling edge occurs to CPU1. The active edge is configured via the WP1 bit in the **PWR control register 4 (PWR_CR4)**.
### 6.6.4 PWR control register 4 (PWR_CR4)

Address offset: 0x00C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

| Bit 31:16 Reserved, must be kept at reset value. |
| Bit 15 **C2BOOT**: Boot CPU2 after reset or wake-up from Stop or Standby modes. |
| 0: CPU2 does not boot after reset or wake-up from Stop or Standby modes. |
| 1: CPU2 boots after wake-up from Stop or Standby modes. |
| Bit 14:10 Reserved, must be kept at reset value. |
| Bit 9 **VBRS**: $V_{BAT}$ battery charging resistor selection |
| 0: Charge $V_{BAT}$ through a 5 kΩ resistor |
| 1: Charge $V_{BAT}$ through a 1.5 kΩ resistor |
| Bit 8 **VBE**: $V_{BAT}$ battery charging enable |
| 0: $V_{BAT}$ battery charging disabled |
| 1: $V_{BAT}$ battery charging enabled |
| Bit 7:5 Reserved, must be kept at reset value. |
| Bit 4 **WP5**: Wakeup pin WKUP5 polarity (available on STM32WB55xx only) |
| This bit defines the polarity used for an event detection on external wake-up pin, WKUP5 |
| 0: Detection on high level (rising edge) |
| 1: Detection on low level (falling edge) |
| Note that this bit is reserved in STM32WB35xx devices. |
| Bit 3 **WP4**: Wakeup pin WKUP4 polarity |
| This bit defines the polarity used for an event detection on external wake-up pin, WKUP4 |
| 0: Detection on high level (rising edge) |
| 1: Detection on low level (falling edge) |
| Bit 2 **WP3**: Wakeup pin WKUP3 polarity (available on STM32WB55xx only) |
| This bit defines the polarity used for an event detection on external wake-up pin, WKUP3 |
| 0: Detection on high level (rising edge) |
| 1: Detection on low level (falling edge) |
| Note that this bit is reserved in STM32WB35xx devices. |
| Bit 1 **WP2**: Wakeup pin WKUP2 polarity (available on STM32WB55xx only) |
| This bit defines the polarity used for an event detection on external wake-up pin, WKUP2 |
| 0: Detection on high level (rising edge) |
| 1: Detection on low level (falling edge) |
| Note that this bit is reserved in STM32WB35xx devices. |
Bit 0  **WP1**: Wakeup pin WKUP1 polarity
This bit defines the polarity used for an event detection on external wake-up pin, WKUP1
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

### 6.6.5  **PWR status register 1 (PWR_SR1)**

Address offset: 0x010

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

<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>WUF1</td>
<td>C2HF</td>
<td>802AF</td>
<td>BLE</td>
<td>CRPEF</td>
<td>802</td>
<td>WUF</td>
<td>802</td>
<td>WUF</td>
<td>802</td>
<td>WUF</td>
<td>BORHF</td>
<td>SMPS</td>
<td>FBF</td>
<td>WUF5</td>
<td>WUF4</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**  Reserved, must be kept at reset value.

- **Bit 15  WUF1**: Internal wake-up interrupt flag
  This bit is set when a wake-up is detected on the internal wake-up line. It is cleared when all internal wake-up sources are cleared.

- **Bit 14  C2HF**: CPU2 Hold interrupt flag
  This bit is set when a CPU2 wake-up is detected when C2BOOT = 0. It is cleared by PWR_SCR.CC2HF.

- **Bit 13  802AF**: 802.15.4 end of activity interrupt flag
  This bit is set when a 802.15.4 activity ends. It is cleared by PWR_SCR.C802AF.

- **Bit 12  BLEAF**: BLE end of activity interrupt flag
  This bit is set when a BLE activity ends. It is cleared by PWR_SCR.CBLEAF.

- **Bit 11  CRPEF**: Enable critical radio phase end of activity interrupt flag
  This bit is set when Radio phase activity ends. It is cleared by PWR_SCR.CCRPEF.

- **Bit 10  802WUF**: 802.15.4 wake-up interrupt flag
  This bit is set when a wake-up is detected on the 802.15.4 line. It is cleared by PWR_SCR.C802WUF.

- **Bit 9  BLEWUF**: BLE wake-up interrupt flag
  This bit is set when a wake-up is detected on the BLE line. It is cleared by PWR_SCR.CBLEWUF.

- **Bit 8  BORHF**: BORH interrupt flag
  This bit is set when the VDD rises above the BORH threshold. It is cleared by PWR_SCR.CBORHF.
Bit 7 **SMPSFBF**: SMPS step-down converter forced in Bypass interrupt flag
This bit is set when the SMPS step-down converter is enabled in SMPS mode and forced in Bypass mode due to the BORH threshold. BORH configuration by BORHC bit in **PWR control register 5 (PWR_CR5)** must select BORH force SMPS step-down converter in Bypass mode.
This bit is cleared by PWR_SCR.CSMPSFBF.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **WUF5**: Wakeup flag 5 (available on STM32WB55xx only)
This bit is set when a wake-up event is detected on wake-up pin, WKUP5. It is cleared by writing ‘1’ in the CWUF5 bit of the PWR_SCR register. Note that this bit is reserved in STM32WB35xx devices.

Bit 3 **WUF4**: Wakeup flag 4
This bit is set when a wake-up event is detected on wake-up pin, WKUP4. It is cleared by writing ‘1’ in the CWUF4 bit of the **PWR status clear register (PWR_SCR)**.

Bit 2 **WUF3**: Wakeup flag 3 (available on STM32WB55xx only)
This bit is set when a wake-up event is detected on wake-up pin, WKUP3. It is cleared by writing ‘1’ in the CWUF3 bit of the **PWR status clear register (PWR_SCR)**. Note that this bit is reserved in STM32WB35xx devices.

Bit 1 **WUF2**: Wakeup flag 2 (available on STM32WB55xx only)
This bit is set when a wake-up event is detected on wake-up pin, WKUP2. It is cleared by writing ‘1’ in the CWUF2 bit of the **PWR status clear register (PWR_SCR)**. Note that this bit is reserved in STM32WB35xx devices.

Bit 0 **WUF1**: Wakeup flag 1
This bit is set when a wake-up event is detected on wake-up pin, WKUP1. It is cleared by writing ‘1’ in the CWUF1 bit of the **PWR status clear register (PWR_SCR)**.

### 6.6.6 **PWR status register 2 (PWR_SR2)**

**Address offset**: 0x014

Reset value: 0x0000 0002. 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>
<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>

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

Bit 14 **PVMO3**: Peripheral voltage monitoring output: $V_{DDA}$ Vs. 1.62 V

0: $V_{DDA}$ voltage is above PVM3 threshold (~ 1.62 V)
1: $V_{DDA}$ voltage is below PVM3 threshold (~ 1.62 V)

**Note**: **PVMO3 is cleared when PVM3 is disabled (PVME3 = 0). After enabling PVM3, the PVM3 output is valid after the PVM3 wake-up time.**
Bit 13  Reserved, must be kept at reset value.

Bit 12  **PVMO1**: Peripheral voltage monitoring output: \( V_{DDUSB} \) vs. 1.2 V
0: \( V_{DDUSB} \) voltage is above PVM1 threshold (\( \sim 1.2 \) V)
1: \( V_{DDUSB} \) voltage is below PVM1 threshold (\( \sim 1.2 \) V)

*Note*: PVMO1 is cleared when PVM1 is disabled (PVME1 = 0). After enabling PVM1, the PVM1 output is valid after the PVM1 wake-up time.

Bit 11  **PVDO**: Programmable voltage detector output
0: \( V_{DD} \) or voltage level on PVD_IN is above the selected PVD threshold
1: \( V_{DD} \) or voltage level on PVD_IN 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 control register 1 (PWR_CR1)**.
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 from 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 a Standby/Shutdown. If the Standby mode is entered while REGLPS bit is still cleared, the wake-up from Standby mode time may be increased.
0: The low-power regulator is not ready
1: The low-power regulator is ready

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

Bit 1  **SMPSF**: SMPS step-down converter SMPS mode flag
This bit indicates that the SMPS step-down converter is in SMPS mode.
0: The SMPS step-down converter is not ready
1: The SMPS step-down converter is ready

Bit 0  **SMPSBF**: SMPS step-down converter Bypass mode flag
This bit indicates that the SMPS step-down converter is in Bypass mode.
0: The SMPS step-down converter is not in Bypass mode (either in Open or SMPS mode)
1: The SMPS step-down converter is in Bypass mode

### 6.6.7 PWR status clear register (PWR_SCR)

Address offset: 0x018

Reset value: 0x0000 0000

Access: Three additional APB cycles are needed to write this register vs. a standard APB write.
### 6.6.8 PWR control register 5 (PWR_CR5)

Address offset: 0x01C

<table>
<thead>
<tr>
<th>Bits 31:15</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 14</td>
<td><strong>CC2HF</strong>: Clear CPU2 Hold interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the C2HF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bit 13</td>
<td><strong>C802AF</strong>: Clear 802.15.4 end of activity interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the 802AF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>CBLEAF</strong>: Clear BLE end of activity interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the BLEAF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>CCRPEF</strong>: Clear critical radio phase end of activity interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the CRPEF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bit 10</td>
<td><strong>C802WUF</strong>: Clear 802.15.4 wake-up interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the 802WUF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bit 9</td>
<td><strong>CBLEWUF</strong>: Clear BLE wake-up interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the BLEWUF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>CBORHF</strong>: Clear BORH interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the SBORHF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bit 7</td>
<td><strong>CSMPSFBF</strong>: Clear SMPS step-down converter forced in Bypass interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the SMPSFBF flag in the PWR_SR1. This bit is always read 0.</td>
</tr>
<tr>
<td>Bits 6:5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 4</td>
<td><strong>CWUF5</strong>: Clear wake-up flag 5 (available on STM32WB55xx only)</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the WUF5 flag in the PWR_SR1 register.</td>
</tr>
<tr>
<td></td>
<td>Note that this bit is reserved in STM32WB35xx devices.</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>CWUF4</strong>: Clear wake-up flag 4</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the WUF4 flag in the PWR_SR1 register.</td>
</tr>
<tr>
<td>Bit 2</td>
<td><strong>CWUF3</strong>: Clear wake-up flag 3 (available on STM32WB55xx only)</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the WUF3 flag in the PWR_SR1 register.</td>
</tr>
<tr>
<td></td>
<td>Note that this bit is reserved in STM32WB35xx devices.</td>
</tr>
<tr>
<td>Bit 1</td>
<td><strong>CWUF2</strong>: Clear wake-up flag 2 (available on STM32WB55xx only)</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the WUF2 flag in the PWR_SR1 register.</td>
</tr>
<tr>
<td></td>
<td>Note that this bit is reserved in STM32WB35xx devices.</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>CWUF1</strong>: Clear wake-up flag 1</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the WUF1 flag in the PWR_SR1 register.</td>
</tr>
</tbody>
</table>
Reset value: 0x0000 427X, where X is factory-programmed. This register is not reset when exiting Standby modes.

Access: Three additional APB cycles are needed to write this register vs. a standard APB write.

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15</td>
<td><strong>SMPSEN</strong>: Enable SMPS step-down converter SMPS mode enabled.</td>
</tr>
<tr>
<td></td>
<td>This bit is reset to 0 when SMPS step-down converter switching on the fly is enabled and the ( V_{DD} ) level drops below the BORH threshold.</td>
</tr>
<tr>
<td></td>
<td>0: SMPS Step down converter SMPS mode disabled.</td>
</tr>
<tr>
<td></td>
<td>1: SMPS Step down converter SMPS mode enabled.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The noise performance of analog signals may be impacted by the integrated SMPS switching. To prevent this the SMPS may be switched off on the fly when measuring analog signals.</td>
</tr>
<tr>
<td>Bit 14:9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>BORHC</strong>: BORH configuration selection</td>
</tr>
<tr>
<td></td>
<td>0: BORH generates a system reset.</td>
</tr>
<tr>
<td></td>
<td>1: BORH forces SMPS step-down converter Bypass mode (BORL still generates a system reset).</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 6:4</td>
<td><strong>SMPSSC</strong>: SMPS step-down converter supply startup current selection</td>
</tr>
<tr>
<td></td>
<td>Startup current is limited to maximum 80 mA + SMPSSC x 20 mA.</td>
</tr>
<tr>
<td>Bit 3:0</td>
<td><strong>SMPSVOS</strong>: SMPS step-down converter voltage output scaling</td>
</tr>
<tr>
<td></td>
<td>These bits are initialized after Option byte loading with factory trimmed value to reach 1.5 V, and can subsequently be overwritten by firmware.</td>
</tr>
<tr>
<td></td>
<td>SMPS step down output voltage step size is 50 mV.</td>
</tr>
<tr>
<td></td>
<td>If factory trimmed value - 0x8 gives 1.50 V on ( V_{FBSMPS} ), to get 1.40 V 0x2 must be subtracted from this value.</td>
</tr>
<tr>
<td></td>
<td>(- 0x0 = ) minimum voltage level</td>
</tr>
<tr>
<td></td>
<td>(- 0xF = ) maximum voltage level</td>
</tr>
<tr>
<td></td>
<td>The factory trimmed value can be obtained from address 0x1FFF 7558 bits [11:8].</td>
</tr>
</tbody>
</table>

### 6.6.9 PWR Port A pull-up control register (PWR_PUCRA)

Address offset: 0x020

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).
### 6.6.10 PWR Port A pull-down control register (PWR_PDCRA)

- **Address offset:** 0x024
- **Reset value:** 0x0000 0000. This register is not reset when exiting Standby modes.
- **Access:** Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 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 |   |   |   |   |   |
| PU15 | PU14 | PU13 | PU12 | PU11 | PU10 | PU9 | PU8 | PU7 | PU6 | PU5 | PU4 | PU3 | PU2 | PU1 | PU0 |   |   |   |   |   |
| 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 **PDy**: Port A pull-down bit y (y=0...15)

When set, this bit activates the pull-down on PA[y] when one of the APC bits is set in **PWR control register 3 (PWR_CR3)** and in **PWR CPU2 control register 3 (PWR_C2CR3)**. The pull-down is not activated if the corresponding PDy bit is also set.

### 6.6.11 PWR Port B pull-up control register (PWR_PUCRB)

- **Address offset:** 0x028
- **Reset value:** 0x0000 0000. This register is not reset when exiting Standby modes.
- **Access:** Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 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 |   |   |   |   |   |
| PU15 | PU14 | PU13 | PU12 | PU11 | PU10 | PU9 | PU8 | PU7 | PU6 | PU5 | PU4 | PU3 | PU2 | PU1 | PU0 |   |   |   |   |   |
| 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 **PUy**: Port A pull-up bit y (y=0...15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in **PWR control register 3 (PWR_CR3)** and in **PWR CPU2 control register 3 (PWR_C2CR3)**.
6.6.12 **PWR Port B pull-down control register (PWR_PDCRB)**

Address offset: 0x02C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

<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>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>

PD15 PD14 PD13 PD12 PD11 PD10 PD9 PD8 PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0
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 **PDy**: Port B pull-down bit y (y=0...15)

When set, this bit activates the pull-down on PB[y] when one of the APC bits is set in PWR control register 3 (PWR_CR3) and in PWR CPU2 control register 3 (PWR_C2CR3). Note that bits 15:10 are reserved in STM32WB35xx devices.

6.6.13 **PWR Port C pull-up control register (PWR_PUCRC)**

Address offset: 0x030

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

<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>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>

PU15 PU14 PU13 PU12 PU11 PU10 PU9 PU8 PU7 PU6 PU5 PU4 PU3 PU2 PU1 PU0
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 **PUy**: Port B pull-up bit y (y=0...15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in PWR control register 3 (PWR_CR3) and in PWR CPU2 control register 3 (PWR_C2CR3). Note that bits 15:10 are reserved in STM32WB35xx devices.
### 6.6.14 PWR Port C pull-down control register (PWR_PDCRC)

Address offset: 0x034

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two 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 **PUy**: Port C pull-up bit y (y=0...15)

When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in **PWR control register 3 (PWR_CR3)** and in **PWR CPU2 control register 3 (PWR_C2CR3)**. The pull-up is not activated if the corresponding PDy bit is also set. Note that bits 13:0 are reserved in STM32WB35xx devices.

### 6.6.15 PWR Port D pull-up control register (PWR_PUCRD) (STM32WB55xx only)

Address offset: 0x038

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two 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  **PUy**: Port D pull-up bit y (y=0...15)
When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in PWR control register 3 (PWR_CR3) and in PWR CPU2 control register 3 (PWR_C2CR3). The pull-up is not activated if the corresponding PDy bit is also set.

### 6.6.16  PWR Port D pull-down control register (PWR_PDCRD)
(STM32WB55xx only)

Address offset: 0x03C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

<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></td>
<td></td>
<td></td>
<td></td>
<td></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  **PDy**: Port D pull-down bit y (y=0...15)
When set, this bit activates the pull-down on PD[y] when one of the APC bits is set in PWR control register 3 (PWR_CR3) and in PWR CPU2 control register 3 (PWR_C2CR3).

### 6.6.17  PWR Port E pull-up control register (PWR_PUCRE)

Address offset: 0x040

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two for a read).
### 6.6.18 PWR Port E pull-down control register (PWR_PDCRE)

Address offset: 0x044

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes.

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

<table>
<thead>
<tr>
<th>Bits 31:5</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 4:0</td>
<td><strong>PUy</strong>: Port E pull-up bit y (y=0...4)</td>
</tr>
<tr>
<td></td>
<td>When set, this bit activates the pull-up on Px[y] when one of the APC bits is set in PWR control register 3 (PWR_CR3) and in PWR CPU2 control register 3 (PWR_C2CR3). The pull-up is not activated if the corresponding PDy bit is also set. Note that bits 3:0 are reserved in STM32WB35xx devices.</td>
</tr>
</tbody>
</table>

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

### 6.6.19 PWR Port H pull-up control register (PWR_PUCRH)

Address offset: 0x058

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

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

<table>
<thead>
<tr>
<th>Bits 31:5</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 4:0</td>
<td><strong>PDy</strong>: Port E pull-down bit y (y=0...4)</td>
</tr>
<tr>
<td></td>
<td>When set, this bit activates the pull-down on PE[y] when one of the APC bits is set in PWR control register 3 (PWR_CR3) and in PWR CPU2 control register 3 (PWR_C2CR3). Note that bits 3:0 are reserved in STM32WB35xx devices.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bits</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>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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>
6.6.20 PWR Port H pull-down control register (PWR_PDCRH)

Address offset: 0x05C

Reset value: 0x0000 0000. This register is not reset when exiting Standby modes and with PWRRST bit in the RCC_APB1RSTR1 register.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two 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>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 31:4 Reserved, must be kept at reset value.

Bit 3 **PD3**: Port H pull-down bit 3
When set, this bit activates the pull-down on PH[y] when one of the APC bits is set in **PWR control register 3 (PWR_CR3)** and in **PWR CPU2 control register 3 (PWR_C2CR3)**.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **PDy**: Port H pull-down bit y (y=0...1) (available on STM32WB55xx only)
When set, this bit activates the pull-down on PH[y] when one of the APC bits is set in **PWR control register 3 (PWR_CR3)** and in **PWR CPU2 control register 3 (PWR_C2CR3)**. Note that bits 1:0 are reserved in STM32WB35xx devices.

6.6.21 PWR CPU2 control register 1 (PWR_C2CR1)

Address offset: 0x080

Reset value: 0x0000 0000. This register is reset after wake-up from Standby mode, except for bits [2:0].
Bits 31:16  Reserved, must be kept at reset value.

Bit 15  **802EWKUP**: 802.15.4 external wake-up signal
  When set this bit forces a wake-up of the 802.15.4 controller. It is automatically reset when
  802.15.4 controller exits its sleep mode.
  0: No action.
  1: Wakeup 802.15.4 controller form its sleep mode.

Bit 14  **BLEEWKUP**: BLE external wake-up
  When set this bit forces a wake-up of the BLE controller. It is automatically reset when BLE
  controller exits its sleep mode.
  0: No action
  1: Wakeup BLE controller from its sleep mode

Bits 13:6  Reserved, must be kept at reset value.

Bit 5  **FPDS**: Flash memory power down mode during LPSleep for CPU2
  This bit selects whether the flash memory is in power down mode or idle mode when both
  CPUs are in Sleep mode. flash memory is set in power down mode only when the system is
  in LPSleep mode and the PWR_CR1.FPDS bit from CPU1 also allows so.
  0: Flash memory in Idle mode when system is in LPSleep mode
  1: Flash memory in power down mode when system is in LPSleep mode

Bit 4  **FPDR**: Flash memory power down mode during LPRun for CPU2
  This bit can only be written to 1 after unlocking this register bit, by first writing (code 0xC1B0)
  into this register (when writing the code register bits are not updated). Selects whether the
  flash memory is in power down mode or idle mode when in LPRun mode. (flash memory can
  only be in power down mode when code is executed from SRAM). Flash memory is set in
  power down mode only when the system is in LPRun mode, and the PWR_CR1.FPDR bit
  from CPU1 also allows so.
  0: Flash memory in Idle mode when system is in LPRun mode
  1: Flash memory in power down mode when system is in LPRun mode

Bit 3  Reserved, must be kept at reset value.
Bits 2:0 **LPMS[2:0]**: Low-power mode selection for CPU2

These bits are not reset when exiting Standby mode.
These bits select the low-power mode entered when CPU2 enters the deepsleep mode. The system low-power mode entered depend also on the PWR_CR1.LPMS allowed low-power mode from CPU1.

- 000: Stop0 mode
- 001: Stop1 mode
- 010: Stop2 mode
- 011: Standby mode
- 1xx: Shutdown mode

Note: If LPR bit is set, Stop2 mode cannot be selected and Stop1 mode must be entered instead of Stop2.

In Standby mode, SRAM2 can be preserved or not, depending on RRS bit configuration in PWR control register 3 (PWR_CR3).

6.6.22 **PWR CPU2 control register 3 (PWR_C2CR3)**

Address offset: 0x084

Reset value: 0x0000 8000. This register is not reset when exiting Standby modes.

Access: Additional APB cycles are needed to access this register vs. those needed for a standard APB access (three for a write and two 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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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.

Bit 15 **EIWUL**: Enable internal wake-up line for CPU2

- 0: Internal wake-up line to CPU2 disabled
- 1: Internal wake-up line to CPU2 enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **APC**: Apply pull-up and pull-down configuration for CPU2

When this bit for CPU2 or the PWR_CR3.APC bit for CPU1 is set, the I/O pull-up and pull-down configurations defined in the PWR_PUCRx and PWR_PDCRx registers are applied. When both bits are cleared, the PWR_PUCRx and PWR_PDCRx registers are not applied to the I/Os.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **E802WUP**: Enable 802.15.4 host wake-up interrupt for CPU2

- 0: Interrupt to CPU2 disabled
- 1: interrupt to CPU2 enabled

Bit 9 **EBLEWUP**: Enable BLE host wake-up interrupt for CPU2

- 0: Interrupt to CPU2 disabled
- 1: interrupt to CPU2 enabled
6.6.23 PWR extended status and status clear register (PWR_EXTSCR)

Address offset: 0x088

Reset value: 0x0000 0000

Access: Three additional APB cycles are needed to write this register vs. 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>
<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>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.

Bit 15 C2DS: CPU2 deepsleep mode
This bit is set by hardware when CPU2 enters deepsleep mode or is hold by C2BOOT.
0: CPU2 is running or in sleep
1: CPU2 is in deepsleep or hold by C2BOOT
Bit 14 **C1DS**: CPU1 deepsleep mode
This bit is set by hardware when CPU1 enters deepsleep mode.
0: CPU1 is running or in sleep
1: CPU1 is in deepsleep

Bit 13 **CRPF**: Critical Radio system phase
This bit is set by hardware when the Radio system wakes up. It is reset either by hardware when the Radio system enters Low-power mode or by software when writing CCRPF.
0: Not a critical Radio system phase
1: Critical Radio system phase ongoing

Bit 12 Reserved, must be kept at reset value.

Bit 11 **C2STOPF**: System Stop flag for CPU2.
This bit is set by hardware and cleared only by any reset or by setting C2CSSF bit.
0: System has not been in Stop mode
1: System has been in Stop mode

Bit 10 **C2SBF**: System Standby flag for CPU2.
This bit is set by hardware and cleared only by a POR reset or by setting C2CSSF bit.
0: System has not been in Standby mode
1: System has been in Standby mode

Bit 9 **C1STOPF**: System Stop flag for CPU1.
This bit is set by hardware and cleared only by any reset or by setting C1CSSF bit.
0: System has not been in Stop mode
1: System has been in Stop mode

Bit 8 **C1SBF**: System Standby flag for CPU1.
This bit is set by hardware and cleared only by a POR reset or by setting C1CSSF bit.
0: System has not been in Standby mode
1: System has been in Standby mode

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

Bit 2 **CCRPF**: Clear critical Radio system phase
Setting this bit clears the CRPF bit.

Bit 1 **C2CSSF**: Clear CPU2 Stop Standby flags
Setting this bit clears the C2STOPF and C2SBF bits.

Bit 0 **C1CSSF**: Clear CPU1 Stop Standby flags
Setting this bit clears the C1STOPF and C1SBF bits.
## 6.6.24 PWR register map and reset value table

### Table 35. PWR 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  | PWR_CR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 1 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | PWR_CR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | PWR_CR3    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | PWR_CR4    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | PWR_SR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | PWR_SR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x018  | PWR_SCR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | PWR_SR5    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | PWR_PUCRA  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | PWR_PDCRA  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x028  | PWR_PUCRB  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x02C  | PWR_PDCRB  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x030  | PWR_PUCRC  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 0  | 0  | 0 | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.2 on page 66 for the register boundary addresses.
7 Peripherals interconnect matrix

7.1 Introduction

Several peripherals have direct connections, enabling autonomous communication and/or synchronization between them. This saves CPU resources and, consequently, reduces power consumption. In addition, these hardware connections remove software latency and result in more predictable system design.

Depending on peripherals, these interconnections can operate in Run, Sleep, Low-power run and sleep, Stop 0, Stop 1 and Stop 2 modes.

7.2 Interconnect matrix implementation

Table 36. Interconnect matrix implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TAMP1</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>RTC_TAMP2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RTC_TAMP3</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

7.3 Connection summary

Table 37. Peripherals interconnect matrix

<table>
<thead>
<tr>
<th>Source</th>
<th>TIM1</th>
<th>TIM2</th>
<th>TIM16</th>
<th>TIM17</th>
<th>LPTIM1</th>
<th>LPTIM2</th>
<th>ADC1</th>
<th>COMP1</th>
<th>COMP2</th>
<th>IRTIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>TIM2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>TIM16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
</tr>
<tr>
<td>TIM17</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</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>
</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>
</tr>
<tr>
<td>ADC1</td>
<td>3</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>8</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>8</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>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HSE</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSE</td>
<td>-</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MSI</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSI</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
7.4 Interconnection details

7.4.1 From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2)

Purpose

Some of the timers are linked together internally for 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 24.3.26: Timer synchronization.

The modes of synchronization are detailed in:

- Section 24.3.26: Timer synchronization for advanced-control timers (TIM1)
- Section 25.3.18: Timers and external trigger synchronization for general-purpose timers (TIM2)

Triggering signals

The output (from Master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1) following a configurable timer event. The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3.

The input and output signals for TIM1 are shown in Figure 144: Advanced-control timer block diagram.

The possible master/slave connections are given in:

- Table 161: TIM1 internal trigger connection
- Table 165: TIM2 internal trigger connection

Active power mode(s)

Run, Sleep, Low-power run, Low-power sleep.
7.4.2 From timer (TIM1/TIM2) and EXTI to ADC (ADC1)

**Purpose**
General-purpose timer TIM2, advanced-control timer TIM1 and EXTI can be used to generate an ADC triggering event.
TIMx synchronization is described in *Section 24.3.27: ADC synchronization*.
ADC synchronization is described in *Section 1.4: 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 signals EXT[15:0] and JEXT[15:0].
The connection between timers and ADC is provided in:
- Table 78: ADC1 - External triggers for regular channels
- Table 79: ADC1 - External trigger for injected channels

**Active power mode(s)**
Run, Sleep, Low-power run, Low-power sleep.

7.4.3 From ADC (ADC1) to timer (TIM1)

**Purpose**
ADC1 can provide trigger event through watchdog signals to advanced-control timers (TIM1).
A description of the ADC analog watchdog setting is provided in *Section 16.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*.
Trigger settings on the timer are provided in *Section 24.3.4: External trigger input*.

**Triggering signals**
The output (from ADC) is on signals ADC_AWDx_OUT n = 1, 2, 3 (for ADC1) x = 1, 2, 3 (three watchdogs on ADC) and the input (to timer) on signal TIMx_ETR (external trigger).

**Active power mode(s)**
Run, Sleep, Low-power run, Low-power sleep.

7.4.4 From HSE, LSE, LSI, MSI, MCO, RTC to timers (TIM2/TIM16/TIM17)

**Purpose**
External clocks (HSE, LSE), internal clocks (LSI, MSI), microcontroller output clock (MCO), GPIO and RTC wakeup interrupt can be used as input to general-purpose timers (TIM16/17) channel 1.
This makes possible calibration of the HSI16/MSI system clocks (with TIM16 and LSE) or of the LSI (with TIM16 and HSE). This is also used to precisely measure LSI (with TIM16 and HSI16) or MSI (with TIM17 and HSI16) oscillator frequency.

When Low Speed External (LSE) oscillator is used, no additional hardware connections are required.

This feature is described in Section 8.2.21: Internal/external clock measurement with TIM16/TIM17.

External clock LSE can be used as input to general-purpose timer (TIM2) on TIM2_ETR pin, see Section 8.2.21: Internal/external clock measurement with TIM16/TIM17..

Active power mode(s)
Run, Sleep, Low-power run, Low-power sleep.

7.4.5 From RTC, COMP1, COMP2 to low-power timers (LPTIM1/LPTIM2)

Purpose
RTC alarm A/B, RTC_TAMP1/2/3 input detection, COMP1/2_OUT can be used as trigger to start LPTIM counters (LPTIM1/2).

Triggering signals
This trigger feature is described in Section 27.4.6: Trigger multiplexer (and following sections).

The input selection is described in Table 173: LPTIM1 external trigger connection and Table 174: LPTIM2 external trigger connection.

Active power mode(s)
Run, Sleep, Low-power run, Low-power sleep, Stop 0, Stop 1, Stop 2 (LPTIM1 only).

7.4.6 From timer (TIM1/TIM2) to comparators (COMP1/COMP2)

Purpose
Advanced-control timer (TIM1) and general-purpose timer (TIM2) can be used as blanking window input to COMP1/COMP2.

The blanking function is described in Section 18.3.7: Comparator output blanking function.

The blanking sources are given in

Section 18.6.1: Comparator 1 control and status register (COMP1_CSR) bits 20:18 BLANKING[2:0]
Section 18.6.2: 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.

Active power mode(s)
Run, Sleep, Low-power run, Low-power sleep.
7.4.7 From USB to timer (TIM2)

Purpose
USB (FS SOF) can generate a trigger to general-purpose timer (TIM2).
Connection of USB to TIM2 is described in Table 165: TIM2 internal trigger connection.

Triggering signals
Internal signal generated by USB_FS Start Of Frame.

Active power mode(s)
Run, Sleep.

7.4.8 From internal analog to ADC1

Purpose
Internal temperature sensor (VTS), internal reference voltage (VREFINT) and VBAT monitoring channel are connected to ADC1 input channel.

This is according to:
- Section 1.2: ADC main features
- Section 1.3.8: Channel selection (CHSEL, SCANDIR)
- Section 1.9: Temperature sensor and internal reference voltage
- Section 1.10: VLCD voltage monitoring

Active power mode(s)
Run, Sleep, low-power run, Low-power sleep.

7.4.9 From comparators (COMP1/COMP2) to timers (TIM1/TIM2/TIM16/TIM17)

Purpose
Comparators (COMP1/COMP2) output values can be connected to timers TIM1/TIM2/TIM16/TIM17 input captures or TIMx_ETR signals.

The connection to ETR is described in Section 24.3.4: External trigger input.

Comparators (COMP1/COMP2) output values can also generate break input signals for timer TIM1 on input pins TIMx_BKIN or TIMx_BKIN2 through GPIO alternate function selection using open drain connection of I/Os.

The possible connections are given in:
- Section 24.4.23: TIM1 option register 1 (TIM1_OR1)
- Section 24.4.28: TIM1 Alternate function register 2 (TIM1_AF2)
- Section 25.4.22: TIM2 option register 1 (TIM2_OR1)
- Section 24.4.27: TIM1 alternate function option register 1 (TIM1_AF1)
- Section 26.2: TIM16/TIM17 main features
Active power mode(s)
Run, Sleep, Low-power run, Low-power sleep.

7.4.10 From system errors to timers (TIM1/TIM16/TIM17)

Purpose
CSS, CPU hard fault, RAM parity error, FLASH ECC double error detection, PVD can generate system errors in the form of timer break toward timers TIM1/TIM16/TIM17.

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

List(s) of possible break source(s) are described in:
- Section 24.3.16: Using the break function (TIM1)
- Section 26.3.11: Using the break function (TIM16/TIM17)
- Figure 253: TIM16/TIM17 block diagram

Active power mode(s)
Run, Sleep, Low-power run, Low-power sleep.

7.4.11 From timers (TIM16/TIM17) to IRTIM

Purpose
General-purpose timer (TIM16/TIM17) output channel TIMx_OC1 are used to generate the waveform of infrared signal output.

The functionality is described in Section 28: Infrared interface (IRTIM).

Active power mode(s)
Run, Sleep, Low-power run, Low-power sleep.
8 Reset and clock control (RCC)

8.1 Reset

There are three types of reset, namely:
1. a system reset
2. a power reset
3. a backup domain reset

8.1.1 Power reset

A power reset is generated when one of the following events occurs:
1. a brown-out reset (BOR)
2. when exiting from Standby mode
3. when exiting from Shutdown mode

A brown-out reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the Backup 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 modes control) are not impacted.

When exiting Shutdown mode, a Brown-out reset is generated, resetting all registers except those in the Backup domain.

8.1.2 System reset

A system reset sets all registers to their reset values, unless specified otherwise in the register description.

A system reset is generated when one of the following events occurs:
1. A low level on the NRST pin (external reset)
2. Window watchdog event (WWDG reset)
3. Independent watchdog event (IWDG reset)
4. A software (SW) reset (see Software reset)
5. Low-power mode security reset (see Low-power mode security reset)
6. Option byte loader reset (see Option byte loader reset)
7. A brown-out reset

The reset source can be identified by checking the reset flags in the RCC control/status register (RCC_CSR).

These sources act on the NRST pin, always kept low during the delay phase. The CPU1 RESET service routine vector is selected via the BOOT0 and BOOT1.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.
In case of an internal reset, the internal pull-up R\textsubscript{PU} is deactivated to reduce power consumption through the pull-up resistor.

**Figure 14. Simplified diagram of the reset circuit**

![Reset Circuit Diagram]

**Software reset**

The SYSRESETREQ bit in CPU1 application interrupt and reset control register may be set to force a software reset on the device (refer to PM0214 “STM32 Cortex®-M4 MCUs and MPUs programming manual”, available on www.st.com).

The SYSRESETREQ bit in CPU2 application interrupt and reset control register may be set to force a software reset on the device.

**Low-power mode security reset**

To prevent that critical applications enter a low-power mode by mistake, two low-power mode security resets are available. If enabled in option bytes, the resets are generated in the following conditions:

1. When entering Standby mode: 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.
2. When entering Stop mode: 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.
3. When entering Shutdown mode: 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: Option bytes description**.

**Option byte loader reset**

The option byte loader reset is generated when the OBL_LAUNCH bit is set in the FLASH_CR register. This bit is used to launch the option byte loading by software.
8.1.3 Backup domain reset

The backup domain has two specific resets.

A backup domain reset is generated when one of the following events occurs:

1. Software reset, triggered by setting the BDRST bit in the RCC backup domain control register (RCC_BDCR).
2. \( V_{DD} \) or \( V_{BAT} \) power on, if both supplies have previously been powered off.

A backup domain reset only affects the LSE oscillator, the RTC, the Backup registers and the RCC Backup domain control register.

8.2 Clocks

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 (high speed internal) 16 MHz RC oscillator clock
- MSI (multispeed internal) RC oscillator clock from 100 kHz to 48 MHz.
- HSE 32 MHz oscillator clock
- PLL clock

The MSI is used as system clock source after startup from Reset, configured at 4 MHz.

The devices have the following additional clock sources:

- LSI1: 32 kHz low speed internal RC, which may drive the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes (shall not be used for RF system Auto-wakeup).
- LSI2: 32 kHz low speed low drift internal RC, which may drive the independent watchdog and optionally the RTC used for Auto-wakeup from Stop and Standby modes.
- LSE: 32.768 kHz low speed external crystal, which optionally drives the RTC used for Auto-wakeup or the RF system Auto-wakeup from Stop and Standby modes, or the real-time clock (RTCLK).
- HSI48: RC 48 MHz internal clock sources to potentially drive the USB Full Speed and the true RNG.

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB frequencies (HCLK1, HCLK2 and HCLK4) the high speed APB (PCLK2) and the low speed APB (PCLK1) domains. The maximum frequency of the AHB (HCLK1 and HCLK4), and of the PCLK1 and PCLK2 domains is 64 MHz. The maximum frequency of the AHB (HCLK2) domain is 32 MHz.
Most peripheral clocks are derived from their bus clock (HCLK, PCLK) except:

- The 48 MHz clock, used for USB FS and true RNG. This clock is derived (selected by software) from one of the following sources:
  - PLL VCO (PLLQCLK only available in Run mode)
  - PLLSAI1 VCO (PLLSAI1QCLK only available in Run mode)
  - MSI clock (only available in Run mode)
  - HSI48 internal oscillator (only available in Run mode)

When the MSI clock is auto-trimmed with the LSE, it can be used by the USB FS device.

The HSI48 clock can be coupled to the clock recovery system allowing adequate clock connection for the USB FS (crystal less solution).

The ADCs clock, which is derived (selected by software) from one of the three following sources:

- system clock (SYSCLK only available in Run mode)
- PLL VCO (PLLAPCLK only available in Run mode)
- PLLSAI1 VCO (PLLSAI1RCLK only available in Run mode)

The UARTs clocks, which are derived (selected by software) from one of the following sources:

- system clock (SYSCLK only available in Run mode)
- HSI16 clock (only available in Run and Stop modes)
- LSE clock (only available in Run and Stop modes)
- APB clock (PCLK depending on which APB is mapped the UART only available in CRun when enabled in UARTxEN and CSleep when also enabled in UARTxSMEN)

The wakeup from Stop mode is supported only when the clock is HSI16 or LSE.

The I2Cs clocks, derived (selected by software) from one of the three following sources:

- system clock (SYSCLK only available in Run mode)
- HSI16 clock (only available in Run and Stop modes)
- APB clock (PCLK depending on which APB is mapped the I2C only available in CRun when enabled in I2CxEN and CSleep when also enabled in I2CxSMEN)

The wakeup from Stop mode is supported only when the clock is HSI.

The SAI1 clock, which is derived (selected by software) from one of the following sources:

- an external clock mapped on SAI1_EXTCLK for SAI1.
- PLL VCO (PLLPCLK only available in Run mode)
- PLLSAI1 VCO (PLLSAI1PCLK only available in Run mode)
- HSI16 clock (only available in Run mode)

The SAI1 can use the HSI16 clock when it is master just to detect an audio activity on the data line, in reception mode. Potentially, it allows to reduce power consumption without having to switch ON the PLL when there is no audio data flow in reception.

The low-power timer (LPTIMx) clocks, which are derived (selected by software) from one of the following sources:

- LSI clock (LSI1 or LSI2 only available in Run and Stop modes)
- LSE clock (only available in Run and Stop modes)
- HSI16 clock (only available in Run mode)
- APB clock (PCLK depending on which APB is mapped the LPTIMx only available in CRun when enabled in LPTIMxEN and CSleep when also enabled in LPTIMxSMEN)
- External clock mapped on LPTIMx_IN1 (only available in Run and Stop modes)

The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE, or in external clock mode.

The RTC and LCD clock, which is derived (selected by software) from one of the three following sources:
- LSE clock
- LSI clock (LSI1 or LSI2)
- HSE clock divided by 32

The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.

The IWDG clock, which is always the LSI clock (LSI1 or LSI2).

The RF system wakeup clock, which is derived (selected by software) from one of the two following sources:
- LSE clock
- HSE clock divided by 1024

The functionality in Stop mode (including wakeup) is supported only when the clock is LSE. When HSE/1024 is selected as RF system wakeup source HSEON bit must be set by the application.

The RF system clock is derived (selected by hardware) from one of the two following sources:
- HSI16 clock
- HSE clock

The functionality in Stop mode is supported only when the clock is HSI16 is selected as wakeup clock by STOPWUCK.

The RCC feeds the CPU1 system timer (SysTick) external clock with the AHB clock (HCLK1) divided by 8. The SysTick can work either with this clock or directly with the CPU1 clock (HCLK1), configurable in the SysTick Control and status register.

FCLK1 acts as CPU1 free-running clock. For more details refer to PM0214.

The RCC feeds the CPU2 system timer (SysTick) external clock with the AHB clock (HCLK2) divided by 8. The SysTick can work either with this clock or directly with the CPU2 clock (HCLK2), configurable in the SysTick Control and status Register.

FCLK2 acts as CPU2 free-running clock.

The clock tree is detailed in Figure 15, where blue is used for system clock, synchronous clock for CPUs and shared buses, and red is used for asynchronous Radio peripheral bus clock.
**Figure 15. Clock tree**

1. LCD is available only on STM32WB55xx devices.
2. For full details about the internal and external clock source characteristics refer to the “Electrical characteristics” section in the device datasheet.
3. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). When the programmable factor is 1, the AHB prescaler must be equal to 1.
8.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources (see Figure 16):

- HSE external crystal
- HSE user external clock

The crystal has to be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance are integrated and can be adjusted according to Frequency tuning.

**Figure 16. HSE clock sources**

<table>
<thead>
<tr>
<th>Clock source</th>
<th>Hardware configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>External clock</td>
<td><img src="image" alt="External clock configuration" /></td>
</tr>
<tr>
<td>Crystal</td>
<td><img src="image" alt="Crystal configuration" /></td>
</tr>
</tbody>
</table>

**External crystal (HSE crystal)**

The 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock, and is mandatory for any Radio operation.

The associated hardware configuration is shown in Figure 16. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag in the *RCC 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 *RCC clock interrupt enable register (RCC_CIER)*.

The HSE Crystal can be switched on and off using the HSEON bit in the *RCC clock control register (RCC_CR)*.

**External source**

In this mode, selectable by setting the HSEON bit in the *RCC clock control register (RCC_CR)*, an external clock source must be provided, with a frequency of 32 MHz. The external clock signal (sinus) with ~45 to 55 % duty cycle (refer to the datasheet) has to drive the OSC_IN pin, while the OSC_OUT pin must be left not connected (see Figure 16).

**Note:** For details on pin availability, refer to the pinout section in the corresponding device datasheet.
Frequency tuning

The HSE oscillator frequency can vary from one chip to another due to manufacturing process variations and used crystal. User can tune the HSE frequency in the application by writing the HSETUNE bits in RCC clock HSE register (RCC_HSECR). The HSE frequency can be measured by outputting the HSE clock on the MCO.

HSE oscillator gain and sense can be controlled by HSEGMC and HSES bits in RCC clock HSE register (RCC_HSECR). Refer to AN5042 for the HSE trimming procedure.

8.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz oscillator.

The HSI16 oscillator has the advantage of providing a clock source at low cost. It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

The HSI16 clock can be selected as system clock after wakeup from Stop modes (Stop0, Stop1 or Stop2), see Section 8.3: Low-power modes. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails, see Section 8.2.11.

When the RF system is enabled the HSI16 must be selected as system clock after wakeup from Stop modes.

The HSI16 clock is used as system clock after restart wakeup from Standby.

Calibration

RC oscillator frequencies can vary from one chip to another because of manufacturing process variations, this is why each device is factory calibrated by ST for 1 % accuracy at $T_A = 25 \degree C$.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the RCC internal clock sources calibration register (RCC_ICSCR).

The RC oscillator speed can be affected by voltage or temperature variations, the user can trim the HSI16 frequency in the application using the HSITRIM bits in the RCC internal clock sources calibration register (RCC_ICSCR).

For more details on how to measure the HSI16 frequency variation, refer to Section 8.2.21.

The HSIRDY flag in the RCC 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 RCC 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 8.2.11.

8.2.3 MSI clock

The MSI clock signal is generated from an internal RC oscillator. Its frequency range can be adjusted by software by using the MSIRANGE[3:0] bits in the RCC clock control register (RCC_CR). Twelve frequency ranges are available: 100 kHz, 200 kHz, 400 kHz, 800 kHz, 1 MHz, 2 MHz, 4 MHz (default value), 8 MHz, 16 MHz, 24 MHz, 32 MHz and 48 MHz.
The MSI clock is used as system clock after restart from Reset, wakeup from Shutdown low-power modes. After restart from Reset, the MSI frequency is set to its default value 4 MHz. Refer to Section 8.3: Low-power modes.

The MSI clock can be selected as system clock after a wakeup from Stop mode (Stop 0, Stop 1 or Stop 2). Refer to Section 8.3: Low-power modes. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to Section 8.2.11.

In addition, when used in PLL-mode with the LSE, it provides a very accurate clock source which can be used by the USB FS device, and feed the PLL to run the system at the maximum speed 64 MHz.

The MSIRDY flag in the RCC clock control register (RCC_CR) indicates if the MSI RC is stable or not. At startup, the MSI RC output clock is not released until this bit is set by hardware. The MSI RC can be switched on and off by using the MSION bit in the RCC clock control register (RCC_CR).

Hardware auto calibration with LSE (PLL-mode)

When a 32.768 kHz external oscillator is present in the application, it is possible to configure the MSI in a PLL-mode by setting the MSIPPLEN bit in the RCC clock control register (RCC_CR). When configured in PLL-mode, the MSI automatically calibrates itself thanks to the LSE. This mode is available for all MSI frequency ranges. At 48 MHz, the MSI in PLL-mode can be used for the USB FS device.

Software calibration

The MSI RC oscillator frequency can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1 % accuracy at 25 °C ambient temperature (TA). After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the RCC internal clock sources calibration register (RCC_ICSCR). If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The MSI frequency in the application can be trimmed by using the MSITRIM[7:0] bits in the RCC_ICSCR register. For more details on how to measure the MSI frequency variation refer to Section 8.2.21.

8.2.4 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator and can be used directly for the USB and for the random number generator (true RNG).

The internal 48 MHz RC oscillator is mainly dedicated to provide a high precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE or an external signal to automatically and quickly adjust the oscillator frequency on-fly. It is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its default frequency which is subject to manufacturing process variations.

For more details on how to configure and use the CRS peripheral refer to Section 40: Clock recovery system (CRS).

The HSI48RDY flag in the Clock recovery register (RCC_CRRRCR) indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched on and off using the HSI48ON bit in the Clock recovery register (RCC_CRRRCR).
8.2.5 PLLs

The device embeds the following PLLs: PLL, PLLSAI1. Each PLL provides up to three independent outputs. The internal PLLs can be used to multiply the HSI, HSE or MSI output clock frequency. The PLLs input frequency must be between 2.66 and 16 MHz. The selected clock source is divided by a programmable factor PLLM from 1 to 8 to provide a clock frequency in the requested input range. Refer to Figure 15: Clock tree and RCC PLL configuration register (RCC_PLLCFGR).

The PLLs 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 PLLxxxON to 0 in RCC clock control register (RCC_CR).
2. Wait until PLLxxxRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLxxxON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, PLLREN in RCC PLL configuration register (RCC_PLLCFGR), or RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR).

An interrupt can be generated when the PLL is ready, if enabled in the RCC clock interrupt enable register (RCC_CIER).

The PLL output frequency must not exceed 64 MHz.

The enable bit of each PLL output clock (PLLPEN, PLLQEN, PLLREN) can be modified at any time without stopping the corresponding PLL. PLLREN cannot be cleared if PLLRCLK is used as system clock.

In Range 2 the PLL M divider input clock shall not exceed 16 MHz. HSEPRE bit in RCC clock control register (RCC_CR) shall select HSE divided by 2.

8.2.6 LSE clock

The LSE crystal is a 32.768 kHz low-speed external 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 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.
The LSE crystal is switched on and off using the LSEON bit in **RCC backup domain control register (RCC_BDCR)**. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the **RCC backup 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 cannot be increased if LSEON = 1.

The LSERDY flag in the **RCC backup 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 **RCC clock interrupt enable register (RCC_CIER)**.

**External source (LSE bypass)**

In this mode, selectable by setting the LSEBYP and LSEON bits in the **RCC AHB1 peripheral clocks enable in Sleep modes register (RCC_AHB1SMENR)** an external clock source must be provided, with a frequency of up to 1 MHz. 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 17**.

**8.2.7 LSI1 clock**

The LSI1 RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG), RTC, LCD and RF wakeup. The clock frequency is 32 kHz. For more details refer to the electrical characteristics section of the datasheet.

The LSI1 RC can be switched on and off using the LSI1ON bit in the **RCC control/status register (RCC_CSR)**.
The LSI1RDY flag in the **RCC control/status register (RCC_CSR)** indicates if the LSI1 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 **RCC clock interrupt enable register (RCC_CIER)**.

### 8.2.8 LSI2 clock

The LSI2 RC acts as a low drift low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG), RTC and LCD wakeup. The clock frequency is ~ 32 kHz. For more details refer to the electrical characteristics section of the datasheets.

The LSI2 RC can be switched on and off using the LSI2ON bit in the **RCC control/status register (RCC_CSR)**.

The LSI2RDY flag in the **RCC control/status register (RCC_CSR)** indicates if the LSI2 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 **RCC clock interrupt enable register (RCC_CIER)**.

After any system reset NRST and before using the LSI2, its oscillator must get the trimming information (LSI2TRIM field in register RCC_SCR).

### 8.2.9 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):
- MSI oscillator
- HSI16 oscillator
- HSE oscillator
- PLLRCLK

The system clock maximum frequency in Range 1 is 64 MHz. After a system reset, the MSI oscillator, at 4 MHz, 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 will occur when the clock source becomes ready. Status bits in the **RCC internal clock sources calibration register (RCC_ICSCR)** indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

When waking up from Standby mode the HSI16 is selected as system clock.

In Range 2 the system clock must not exceed 16 MHz. HSEPRE bit in **RCC clock control register (RCC_CR)** shall select HSE divided by 2.

### 8.2.10 Clock source frequency versus voltage scaling

*Table 38* gives the different clock source frequencies depending on the product voltage range.
8.2.11 Clock security system (CSS) on HSE

The 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 the advanced-control timers (TIM1 and TIM16/17) and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the CPU1 and CPU2 NMI (non-maskable interrupt) exception vector.

Note: Once the HSE CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and a NMI is automatically generated. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR user must clear the CSS interrupt by setting the CSSC bit in the RCC clock interrupt clear register (RCC_CICR).

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the MSI or the HSI16 oscillator depending on the STOPWUCK configuration in the RCC clock configuration register (RCC_CFGR), and the disabling of the HSE oscillator. If the HSE clock (divided or not) is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

8.2.12 Clock security system on LSE (LSECSS)

The clock security system on LSE can be activated by software writing the LSECSSON bit in the RCC control/status register (RCC_CSR). This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSI1ON enabled) and ready (LSERDY and LSIRDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes except VBAT. It is working also under system reset (excluding power on reset). If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC but no hardware action is made to the registers. If the MSI was in PLL-mode, this mode is disabled.

In Standby mode a wakeup is generated. In other modes an interrupt can be sent to wakeup the software (see RCC clock interrupt enable register (RCC_CIER), RCC clock interrupt flag register (RCC_CIFR), RCC clock interrupt clear register (RCC_CICR)).

The software MUST then disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), and change the RTC clock source (no clock or LSI or HSE, with RTCSEL), or take any required action to secure the application.

Table 38. Maximum clock source frequency

<table>
<thead>
<tr>
<th>Product voltage range</th>
<th>Clock frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSI</td>
</tr>
<tr>
<td>Range 1</td>
<td>48 MHz</td>
</tr>
<tr>
<td>Range 2</td>
<td>16 MHz range</td>
</tr>
</tbody>
</table>

1. HSEPRE shall select divide by 2.
8.2.13 LSI source selection

The LSI used in the system can be selected to come either from LSI1 or LSI2. Whenever LSI2 is turned on by LSI2ON register bit, the LSI2 (when ready) is selected as LSI source.

To switch from LSI2 to LSI1 without interrupting the LSI clock, the LSI1 must first be switched on by LSI1ON register bit. The FW must verify that the LSI1 is ready by LSI1RDY register bit before disabling LSI2 in LSI2ON register bit.

8.2.14 SMPS step-down converter clock

The SMPS step-down converter clock source can be either the MSI, HSI16 or HSE clock. It is selected by programming the SMPSSEL in the RCC SMPS step-down converter control register (RCC_SMPSCR). The system must always be configured so as to get a SMPS step-down converter clock frequency between 4 and 8 MHz. The appropriate division factor shall be set in SMPSDIV. There is a further fixed division by 2 before driving the clock to the SMPS step-down converter.

After a system reset, the HSI16 is selected as SMPS step-down converter clock.

When entering low power modes Stop1, Stop2, Standby, or Shutdown the SMPS step-down converter clock source selected in SMPSSEL in the RCC SMPS step-down converter control register (RCC_SMPSCR) must be set to select the same source as the system clock in SW in the RCC clock configuration register (RCC_CFGR). In Stop mode the clock source selected in SMPSSEL in RCC SMPS step-down converter control register (RCC_SMPSCR) must be set to select the same source as the system clock in STOPWUCK in RCC clock configuration register (RCC_CFGR).

When waking up from Standby mode and powering on the VCORE supply, the HSI16 is selected as SMPS step-down converter clock, independent from the selection in SMPSSEL.

When the Radio system is active the HSE is selected as SMPS step-down converter clock, independent from the selection in SMPSSEL.

The SMPS step-down converter clock source used is indicated by the SMPSSWS in RCC SMPS step-down converter control register (RCC_SMPSCR).

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay). If a not yet ready clock source is selected, the switch occurs only when the clock source becomes ready. Status bits in the RCC internal clock sources calibration register (RCC_ICSCR) indicate which clock(s) is (are) ready, and the SMPSSWS bit indicates the clock currently used as a SMPS step-down converter clock.

The division factor in SMPSDIV depends upon the selected SMPS step-down converter clock source in SMPSSEL. When MSI is selected in SMPSSEL, The MSIRANGE is also taken into account to determine the selected MSI frequency. This allows to switch from one clock source to another, by keeping the same SMPS step-down converter clock frequency. The MSI can be selected as SMPS step-down converter clock source only when a supported SMPS step-down converter clock MSIRANGE is set.

When the RF system is active, the HW selects automatically the HSE source with the current SMPSDIV division ratio.
8.2.15 ADC clock

The ADC clock is derived from the system clock, or from the PLL or PLLSAI1 output. It can reach 64 MHz and can be divided by the prescaler 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’.

8.2.17 Timer clock

The timer clock frequencies are automatically defined by hardware. There are two cases:

1. If the APB prescaler equals 1, the timer clock frequencies are set to the same frequency as that of the APB domain.
2. Otherwise, they are set to twice (×2) the frequency of the APB domain.

---

**Table 39. SMPS step-down converter clock source selection and division**

<table>
<thead>
<tr>
<th>SMPSSSEL</th>
<th>HSI16</th>
<th>HSE</th>
<th>MSI</th>
<th>HSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMPSDIV</td>
<td>16 MHz</td>
<td>16 MHz</td>
<td>24 MHz</td>
<td>32 MHz</td>
</tr>
<tr>
<td>00</td>
<td>1 (8 MHz)</td>
<td>1 (8 MHz)</td>
<td>Reserved</td>
<td>2 (8 MHz)</td>
</tr>
<tr>
<td>01</td>
<td>2 (4 MHz)</td>
<td>2 (4 MHz)</td>
<td>3 (4 MHz)</td>
<td>4 (4 MHz)</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.2.16 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 RCC backup domain control register (RCC_BDCR). This selection cannot be modified without resetting the Backup 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 Backup 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 VDD supply is switched off, provided the VBAT supply is maintained.
- If LSI is selected as the RTC clock the RTC state is not guaranteed if the VDD 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 VDD supply is powered off or if the internal voltage regulator is powered off (removing power from the V_CORE domain).

When the RTC clock is LSE or LSI, the RTC remains clocked and functional under system reset.
### 8.2.18 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI clock is forced on.

When neither the LSI1 oscillator nor the LSI2 oscillator is enabled when starting the IWDG, the LSI1 oscillator is forced on. After the LSI1 oscillator temporization, the clock is provided to the IWDG.

When LSI2 is switched on with the LSI2ON bit, the LSI clock is switched from LSI1 to LSI2.

When LSI2 is disabled by LSI2ON (when the IWDG is running), the LSI1 oscillator is forced on. The LSI clock is switched to LSI1 when ready, and only then LSI2 is stopped.

### 8.2.19 True RNG clock

The true random number generator (RNG) seed clock is derived from the same clock selected for the USB, LSI or LSE clock.

### 8.2.20 Clock-out capability

**MCO**

The microcontroller clock output (MCO) capability enables the clock to be output on the external MCO pin. One of the following clock signals can be selected as MCO clock:

- LSI1 (available in Run and Stop modes)
- LSI2 (available in Run and Stop modes)
- LSE (available in Run and Stop modes)
- MSI (available in Run mode)
- HSI (available in Run mode, when enabled by HSION)
- HSI48 (available in Run mode)
- HSE (available in Run mode)
- PLLRCLK (available in Run mode)
- SYSCLK (available in Run mode)

The selection is controlled by the MCOSEL[3:0] bits of the **RCC clock configuration register (RCC_CFGR)**. The selected clock can be divided with the MCOPRE[2:0] field of the **RCC clock configuration register (RCC_CFGR)**.

The clock on MCO is available only in Run down to Stop1 mode. Low frequency clocks (LSIx, LSE) are available down to Stop1 mode. The clock MCO is not available in Stop2, Standby, and Shutdown modes.

**LSCO**

The LSCO output enables a low speed clock to be output on the external LSCO pin:

- LSI (LSI1 or LSI2)
- LSE

The selection is controlled by the LSCOSEL, and enabled with the LSCOEN in the **RCC backup domain control register (RCC_BDCR)**.

The clock on LSCO is available in Run, Stop, and on one GPIO in Standby and Shutdown modes.

The configuration registers of the corresponding GPIO port must be programmed in alternate function mode.
8.2.21 Internal/external clock measurement with TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM16 or TIM17 channel 1 input capture, as shown on Figure 18 and Figure 19.

**Figure 18. Frequency measurement with TIM16 in capture mode**

The input capture channel of TIM16 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP[1:0] bits in the TIM16_OR register. The possibilities are the following ones:

- TIM16 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM16 Channel1 is connected to the LSI clock.
- TIM16 Channel1 is connected to the LSE clock.
- TIM16 Channel1 is connected to the RTC wakeup interrupt signal. In this case the RTC interrupt should be enabled.

**Figure 19. Frequency measurement with TIM17 in capture mode**

The input capture channel of the Timer 17 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1_RMP[1:0] bits in the TIM17_OR register. The possibilities are the following ones:

- TIM17 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheet.
- TIM17 Channel1 is connected to the MSI clock.
- TIM17 Channel1 is connected to the HSE/32 clock.
- TIM17 Channel1 is connected to the microcontroller clock output (MCO), this selection is controlled by the MCOSEL[3:0] bits of the Clock configuration register (RCC_CFGR).
Calibration of the HSI16 and the MSI

For TIM16, the primary purpose of connecting the LSE to the channel 1 input capture is to be able to precisely measure the HSI16 and MSI system clocks (for this, either the HSI16 or MSI should be used as the system clock source). The number of HSI16 (MSI, respectively) clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few ppm tens), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process- and/or temperature- and voltage-related frequency deviations.

Both the MSI and HSI16 oscillators have dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (e.g. the HSI/LSE ratio): the precision is therefore closely related to the ratio between the two clock sources (the higher the ratio, the better the measurement).

If LSE is not available, HSE32 is the better option to reach the most precise possible calibration.

It is however impossible to have a good enough resolution when the MSI clock is low (typically below 1 MHz). In this case, it is advised to:

- accumulate the results of several captures in a row
- use the timer input capture prescaler (up to one capture every eight periods)
- use the RTC wakeup interrupt signal (when the RTC is clocked by the LSE) as the input for the channel1 input capture. This improves the measurement precision. For this purpose the RTC wakeup interrupt must be enable.

Calibration of the LSI

The calibration of the LSI will follow the same pattern that for the HSI, but changing the reference clock. It is necessary to connect LSI clock to the channel 1 input capture of the TIM16. Then define the HSE as system clock source, the number of his clock counts between consecutive edges of the LSI signal provides a measure of the internal low speed clock period.

The basic concept consists in providing a relative measurement (e.g. the HSE/LSI ratio). The precision is therefore closely related to the ratio between the two clock sources, higher ratios result in more accurate measurements.

8.2.22 Peripheral clocks enable

Most peripheral bus and kernel clocks can individually be enabled per CPU. The RCC_AHBxENR, and RCC_APBxENRy enable peripheral clocks for CPU1 and RCC_C2_AHBxENR, and RCC_C2_APBxENR for CPU2. The peripheral clocks will follow the CPU(s) state for which they are enabled, see Table 40.

Peripheral bus clock activity during the CPU Sleep mode is controlled by the peripheral clock CPU sleep mode enable bit of the RCC_AHBxSMENR, and RCC_APBxSMENRy for CPU1 and RCC_C2_AHBxSMENR, and RCC_C2_APBxSMENRy for CPU2 registers. The peripheral bus clock during Sleep mode will follow the CPU(s) state for which is it enabled, see Table 40.
When the peripheral bus clock is not active, the peripheral registers read or write accesses are not supported.

When the peripheral bus clock is active the peripheral can be accessed by both CPUs regardless of which CPU has enabled the peripheral bus clock in its CPU xxxEN bit. However, when the peripheral bus clock is enabled by only one CPU, and this CPU enters low power mode, the peripheral bus clock is stopped (also depending on this CPUs xxxSMEN setting) and peripheral access for the other CPU is no longer supported. It is therefore good practice to enable the peripheral bus clock with the CPU dedicated clock enable.

When the peripheral kernel clock is not active, the peripheral functionality is stopped. 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 two clock cycles delay before the clock becomes active.

Caution: Just after enabling the clock for a peripheral, software must wait for a delay before accessing the peripheral registers.

Note: The BLE-IP when active, will activate the BLE bus and SRAM2 bus interface clocks.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>xxxEN</th>
<th>xxxSMEN</th>
<th>CPU mode</th>
<th>System mode</th>
<th>Bus clock</th>
<th>Kernel clock(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td></td>
<td>Any</td>
<td>Any</td>
<td>Stopped</td>
<td>Stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CRun</td>
<td>Run</td>
<td>Clocked</td>
<td>Clocked</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>CSleep and CStop</td>
<td>Run</td>
<td>Stopped</td>
<td>Clocked</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>CStop</td>
<td>Stop</td>
<td>Stopped</td>
<td>Clocked when selected from</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– HSI16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– LSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– LSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Stopped when selected from</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– bus clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– SYSCLK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– PLL clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– MSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– HSI48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>– PLLSAI1 clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Standby or Shutdown</td>
<td>Stopped</td>
<td>Stopped</td>
<td>Stopped</td>
</tr>
</tbody>
</table>

1. Only the peripherals SAI, I2C, LPTIM, USART, LPUART, USB FS, true RNG and ADC have a kernel clock.
8.3 Low-power modes

AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
Sleep and Low-power sleep modes stop the CPU clock. The memory interface clocks (Flash and SRAM1 and SRAM2 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, Stop 1 and Stop 2) stops most clocks in the VCORE domain and disables the PLLs, the MSI and the HSE oscillators. The HSI16 may be kept running when requested by the IPs (USART1, LPUART1, I2C1, I2C3) that make it possible to wakeup from Stop modes.

All U(S)ARTs, LPUARTs and I^2Cs have the capability to enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as the clock source for that peripheral).

All U(S)ARTs, LPUARTs and LPTIMs can also be driven by 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). In that case the LSE remains always ON in Stop mode (they do not have the capability to turn on the LSE oscillator).
The LPTIMs can also be driven by the LSI oscillator when the system is in Stop mode (if LSI is selected as clock source for that peripheral) and the LSI oscillator is enabled (LSI1ON or LSI2ON).

Standby and Shutdown modes stops all the clocks in the VCORE domain and disables the PLL, the HSI, the MSI and the HSE oscillators.

The low power modes mode can be overridden for debugging by setting the DBG_SLEEP, DBG_STOP or DBG_STANDBY bits in the DBGMCU.CR register. In addition the EXTI CDBGPWRUPREQ events can be used to allow debugging in Stop modes (see Table 41).

<table>
<thead>
<tr>
<th>Mode</th>
<th>CDBGPWRUPREQ</th>
<th>DBGMCU</th>
<th>Debug</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU1</td>
<td>DBG_STANDBY</td>
<td>DBG_STOP</td>
</tr>
<tr>
<td>Sleep</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Stop0 and Stop1</td>
<td>Disabled</td>
<td></td>
<td>Disabled</td>
</tr>
<tr>
<td></td>
<td>Enabled</td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>Stop0, Stop1 and Stop2</td>
<td>X</td>
<td>Disabled</td>
<td>-</td>
</tr>
<tr>
<td>Standby</td>
<td>X</td>
<td>Disabled</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = Do not use.

When leaving the Stop modes (Stop0, Stop1 or Stop2), the system clock is either MSI or HSI, depending on the software configuration of the STOPWUCK bit in the RCC_CFGR register. When the RF system is enabled the HSI16 shall be selected as in STOPWUCK. When STOPWUCK select the HSI16 clock when leaving Stop mode, the C2HPRE is reset. to select divide by 1. The frequency (range and user trim) of the MSI oscillator is the one configured before entering Stop mode. The user trim of HSI16 is kept. If the MSI was in PLL-mode before entering Stop mode, the PLL-mode stabilization time must be waited for after
wakeup even if the LSE was kept ON during the Stop mode. Before entering Stop 0 mode
with SMPS step-down converter enabled the HSI16 must be kept on by setting the
HSIKERON register bit.

When leaving the Standby mode, the system clock is HSI.

When leaving the Shutdown modes, the system clock is MSI. The MSI frequency at wakeup
from Shutdown mode is 4 MHz. The user trim is lost.

If a Flash memory programming operation is on going, Stop, Standby and Shutdown modes
entry is delayed until the Flash memory interface access is finished. If an access to the APB
domain is ongoing, Stop, Standby and Shutdown modes entry is delayed until the APB
access is finished.
8.4 RCC registers

8.4.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0061 (after POR reset), 0x0000 0160 (after wakeup from Standby reset)

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>r</td>
<td>rw</td>
<td>r</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>
<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.

Bit 27 PLLSA1RDY: SAI PLL clock ready flag
Set by hardware to indicate that the PLLSAI1 is locked.
0: PLLSAI1 unlocked
1: PLLSAI1 locked

Bit 26 PLLSA1ON: SAI PLL enable
Set and cleared by software to enable PLLSAI1.
Cleared by hardware when entering Stop, Standby or Shutdown mode.
0: PLLSAI1 OFF
1: PLLSAI1 ON

Bit 25 PLLRDY: System PLL clock ready flag
Set by hardware to indicate that the system PLL is locked.
0: PLL unlocked
1: PLL locked

Bit 24 PLLON: System 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

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 HSEP: HSE system clock and PLL M divider prescaler
Set and cleared by software to control the division factor of the system clock and PLL M divider input when selecting HSE clock.
0: SYSCLK and PLL M divider input clocks are not divided (HSE)
1: SYSCLK and PLL M divider input clocks are divided by 2 (HSE/2)
Bit 19 **CSSON**: HSE clock security system enable
Set by software to enable the clock security system. When CSSON is set, the HSE lock 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: HSE clock security system OFF (clock detector OFF)
1: HSE clock security system ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 Reserved, must be kept at reset value.

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 six HSE clock cycles. Regardless of the value of the HSEON bit, the HSE oscillator starts on RF demand when needed. It is possible to have HSEON = 0 and HSERDY = 1.*

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 15:13 Reserved, must be kept at reset value.

Bit 12 **HSIKERDY**: HSI16 kernel clock ready flag for peripherals requests.
Set by hardware to indicate that HSI16 oscillator is stable when enabled by HSIKERON or a peripheral kernel clock request. Not set when HSI16 is enabled by software by setting HSION, or by wakeup from Standby.
0: HSI16 oscillator not ready
1: HSI16 oscillator ready

Bit 11 **HSIASFS**: HSI16 automatic start from Stop
Set and cleared by software. When the system wakeup clock is MSI, this bit is used to wakeup the HSI16 is parallel of the system wakeup.
0: HSI16 oscillator is not enabled by hardware when exiting Stop mode with MSI as wakeup clock.
1: HSI16 oscillator is enabled by hardware when exiting Stop mode with MSI as wakeup clock.

Bit 10 **HSIRDY**: HSI16 clock ready flag. (After wakeup from Standby this bit is read ‘b1 once the HSI16 is ready)
Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION, or by wakeup from Standby. Not set when HSI16 is enabled by HSIKERON or by IP request.
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 kernel clocks.
Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 enabled by
HSIKERON can only feed USARTs, LPUARTs and I²Cs peripherals configured with HSI16 as
kernel clock. Keeping the HSI16 ON in Stop mode avoids slowing 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.
Set by hardware to force the HSI16 oscillator ON when STOPWUCK = 1 or HSIASFS = 1
when leaving Stop modes, or in case of failure of the HSE crystal oscillator.
This bit is set by hardware if the HSI16 is used directly or indirectly as system clock.
0: HSI16 oscillator OFF
1: HSI16 oscillator ON

Bits 7:4 **MSIRANGE[3:0]**: MSI clock ranges
These bits are configured by software to choose the frequency range of MSI when
MSIRANGE is set. Twelve frequency ranges are available:
0000: range 0, ~100 kHz
0001: range 1, ~200 kHz
0010: range 2, ~400 kHz
0011: range 3, ~800 kHz
0100: range 4, ~1 MHz
0101: range 5, ~2 MHz
0110: range 6, ~4 MHz (reset value)
0111: range 7, ~8 MHz
1000: range 8, ~16 MHz
1001: range 9, ~24 MHz
1010: range 10, ~32 MHz
1011: range 11, ~48 MHz
others: not allowed (hardware write protection)

Note: **Warning**: MSIRANGE can be modified when MSI is OFF (MSION = 0) or when MSI is
ready (MSIRDY = 1). MSIRANGE must NOT be modified when MSI is ON and NOT
ready (MSION = 1 and MSIRDY = 0).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **MSIPLLEN**: MSI clock PLL enable
Set and cleared by software to enable/ disable the PLL part of the MSI clock source.
MSIPLLEN must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set
by hardware). There is a hardware protection to avoid enabling MSIPLLEN if LSE is not
ready.
This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the clock security
system on LSE detects a LSE failure (refer to RCC_CSR register).
0: MSI PLL OFF
1: MSI PLL ON

Bit 1 **MSIRDY**: MSI clock ready flag (After reset this bit is read 'b1 once the MSI is ready)
This bit is set by hardware to indicate that the MSI oscillator is stable.
0: MSI oscillator not ready
1: MSI oscillator ready

Note: **Once the MSION bit is cleared, MSIRDY goes low after 6 MSI clock cycles.**
Bit 0 **MSION**: MSI clock enable
This bit is set and cleared by software.
Cleared by hardware to stop the MSI oscillator when entering Stop, Standby or Shutdown mode.
Set by hardware to force the MSI oscillator ON when exiting Standby or Shutdown mode.
Set by hardware to force the MSI oscillator ON when STOPWUCK = 0 when exiting from Stop modes, or in case of a failure of the HSE oscillator
Set by hardware when used directly or indirectly as system clock.
0: MSI oscillator OFF
1: MSI oscillator ON

### 8.4.2 RCC internal clock sources calibration register (RCC_ICSCR)
Address offset: 0x004
Reset value: 0x40XX 00XX (X is factory-programmed)
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>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>
</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 30:24 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.
The default value is 64, which, when added to the HSICAL value, should trim the HSI16 to 16 MHz ± 1 %.

**Bits 23:16 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.

**Bits 15:8 MSITRIM[7:0]**: MSI clock trimming
These bits provide an additional user-programmable trimming value that is added to the MSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the MSI.
The default value is 0, which, when added to the MSICAL value, should trim the MSI to its mid frequency.

**Bits 7:0 MSICAL[7:0]**: MSI clock calibration
These bits are initialized at startup with the factory-programmed MSI calibration trim value. When MSITRIM is written, MSICAL is updated with the sum of MSITRIM and the factory trim value.
8.4.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x008

Reset value: 0x0007 0000 (after POR reset), 0x0007 0001 (after wakeup from Standby)

Access: 0 wait state 2 word, half-word and byte access

One or two 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>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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]:** Microcontroller clock output prescaler

These bits are set and cleared by software.

- 000: MCO is divided by 1
- 001: MCO is divided by 2
- 010: MCO is divided by 4
- 011: MCO is divided by 8
- 100: MCO is divided by 16
- Others: not allowed

Bits 27:24 **MCOSEL[3:0]:** Microcontroller clock output

Set and cleared by software.

- 0000: MCO output disabled, no clock on MCO
- 0001: SYSCLK system clock selected
- 0010: MSI clock selected.
- 0011: HSI16 clock selected.
- 0100: HSE clock selected (after stabilization, after HSERDY = 1)
- 0101: Main PLLRCLK clock selected
- 0110: LSI1 clock selected
- 0111: LSI2 clock selected
- 1000: LSE clock selected
- 1001: Internal HSI48 clock selected
- 1100: HSE clock selected (before stabilization, after HSEON = 1)
- Others: Reserved

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:19 Reserved, must be kept at reset value.
Bit 18 **PPRE2F**: PCLK2 prescaler flag (APB2)
- Set and reset by hardware to acknowledge PCLK2 prescaler programming
- Reset when a new prescaler value is programmed in PPRE2. set when the programmed value is actually applied.
  - 0: PCLK2 prescaler value not yet applied
  - 1: PCLK2 prescaler value applied

Bit 17 **PPRE1F**: PCLK1 prescaler flag (APB1)
- Set and reset by hardware to acknowledge PCLK1 prescaler programming
- Reset when a new prescaler value is programmed in PPRE1. set when the programmed value is actually applied.
  - 0: PCLK1 prescaler value not yet applied
  - 1: PCLK1 prescaler value applied

Bit 16 **HPREF**: HCLK1 prescaler flag (CPU1, AHB1, AHB2, AHB3 and SRAM1)
- Set and reset by hardware to acknowledge HCLK1 prescaler programming
- Reset when a new prescaler value is programmed in HPRE. set when the programmed value is actually applied.
  - 0: HCLK1 prescaler value not yet applied
  - 1: HCLK1 prescaler value applied

Bit 15 **STOPWUCK**: Wakeup from Stop and CSS backup clock selection
- Set and cleared by software to select the system clock used when exiting Stop mode.
- The selected clock is also used as emergency clock for the clock security system on HSE.
- Warning: STOPWUCK must not be modified when the HSE clock security system is enabled by CSSON in **RCC clock control register (RCC_CR)** register and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW= 10).
  - 0: MSI oscillator selected as wakeup from stop clock and CSS backup clock.
  - 1: HSI16 oscillator selected as wakeup from stop clock and CSS backup clock

Bit 14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]**: PCLK2 high-speed prescaler (APB2)
- Set and cleared by software to control the division factor of the PCLK2 clock (APB2).
  - The PPRE2F flag can be checked to know if the programmed PPRE2 prescaler value is applied.
    - 0xx: HCLK1 not divided
    - 100: HCLK1 divided by 2
    - 101: HCLK1 divided by 4
    - 110: HCLK1 divided by 8
    - 111: HCLK1 divided by 16

Bits 10:8 **PPRE1[2:0]**: PCLK1 low-speed prescaler (APB1)
- Set and cleared by software to control the division factor of the PCLK1 clock (APB1).
  - The PPRE1F flag can be checked to know if the programmed PPRE1 prescaler value is applied.
    - 0xx: HCLK1 not divided
    - 100: HCLK1 divided by 2
    - 101: HCLK1 divided by 4
    - 110: HCLK1 divided by 8
    - 111: HCLK1 divided by 16
Bits 7:4 **HPRE[3:0]**: HCLK1 prescaler (CPU1, AHB1, AHB2, AHB3 and SRAM1.)
Set and cleared by software to control the division factor of the HCLK1 clock (CPU1, AHB1, AHB2, AHB3 and SRAM1.).
The HPREF flag can be checked to know if the programmed HPRE prescaler value is applied.

**Caution:** Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to **Section 6.1.6: Dynamic voltage scaling management**). After a write operation to these bits and before decreasing the voltage range the register bit HPREF must be read to be sure that the new value has been taken into account.

- 0001: SYSCLK divided by 3
- 0010: SYSCLK divided by 5
- 0101: SYSCLK divided by 6
- 0110: SYSCLK divided by 10
- 0111: SYSCLK divided by 32
- 1000: SYSCLK divided by 2
- 1001: SYSCLK divided by 4
- 1010: SYSCLK divided by 8
- 1011: SYSCLK divided by 16
- 1100: SYSCLK divided by 64
- 1101: SYSCLK divided by 128
- 1110: SYSCLK divided by 256
- 1111: SYSCLK divided by 512
- Others: SYSCLK not divided

Bits 3:2 **SWS[1:0]**: System clock switch status
Set and cleared by hardware to indicate which clock source is used as system clock.
- 00: MSI oscillator used as system clock
- 01: HSI16 oscillator used as system clock
- 10: HSE used as system clock
- 11: PLL used as system clock

Bits 1:0 **SW[1:0]**: System clock switch
Set and cleared by software to select system clock source (SYSCLK).
- Configured by HW to force MSI oscillator selection when exiting Shutdown mode.
- Configured by HW to force HSI16 oscillator selection when exiting Standby mode.
- Configured by HW to force MSI or HSI16 oscillator selection when exiting Stop mode or in case of failure of the HSE oscillator, depending on STOPWUCK value.
- 00: MSI selected as system clock
- 01: HSI16 selected as system clock
- 10: HSE selected as system clock
- 11: PLL selected as system clock

**Note:** The latency when changing SW and SWS bits is set according to the new configuration when switching the SYSCLK clock source, in a maximum of three cycles of the previous clock.

### 8.4.4 RCC PLL configuration register (RCC_PLLCFGR)

Address offset: 0x00C
Reset value: 0x2204 0100
Access: No wait state, word, half-word and byte access

This register is used to configure the PLL clock outputs according to the following formulas:

\[ f(VCO \text{ clock}) = f(PLL \text{ clock input}) \times \left( \frac{PLLN}{PLLM} \right) \]

\[ f(PLL_P) = \frac{f(VCO \text{ clock})}{PLL_P} \]

\[ f(PLL_Q) = \frac{f(VCO \text{ clock})}{PLL_Q} \]

\[ f(PLL_R) = \frac{f(VCO \text{ clock})}{PLL_R} \]

<table>
<thead>
<tr>
<th>Bits 31:29</th>
<th>PLLR[2:0]: Main PLL division factor for PLLRCLK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and cleared by software to control the frequency of the main PLL output clock PLLRCLK. This output can be selected as system clock. These bits can be written only if the PLL is disabled.</td>
</tr>
<tr>
<td></td>
<td>PLLRCLK output clock frequency = VCO frequency / PLLR with PLLR = 2, 3, 4, ..., or 8 [VCO frequency / (N + 1)]</td>
</tr>
<tr>
<td></td>
<td>000: reserved</td>
</tr>
<tr>
<td></td>
<td>001: PLLR = 2</td>
</tr>
<tr>
<td></td>
<td>010: PLLR = 3</td>
</tr>
<tr>
<td></td>
<td>011: PLLR = 4</td>
</tr>
<tr>
<td></td>
<td>100: PLLR = 5</td>
</tr>
<tr>
<td></td>
<td>101: PLLR = 6</td>
</tr>
<tr>
<td></td>
<td>110: PLLR = 7</td>
</tr>
<tr>
<td></td>
<td>111: PLLR = 8</td>
</tr>
<tr>
<td></td>
<td>The software has to set these bits so that 64 MHz is not exceeded on this domain.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>PLLREN: Main PLL PLLRCLK output enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software to enable the PLLRCLK output of the main PLL (used as system clock).</td>
</tr>
<tr>
<td></td>
<td>This bit cannot be written when PLLRCLK output of the PLL is used as System clock.</td>
</tr>
<tr>
<td></td>
<td>To save power, when the PLLRCLK output of the PLL is not used, the value of PLLREN should be 0.</td>
</tr>
<tr>
<td></td>
<td>0: PLLRCLK output disabled</td>
</tr>
<tr>
<td></td>
<td>1: PLLRCLK output enabled</td>
</tr>
</tbody>
</table>
Bits 27:25 **PLLQ[2:0]**: Main PLL division factor for PLLQCLK
Set and cleared by software to control the frequency of the main PLL output clock PLLQCLK. This output can be selected for USB and true RNG clock. These bits can be written only if PLL is disabled.

PLLQCLK output clock frequency = VCO frequency / PLLQ = 2, 3, 4,... or 8 [VCO frequency / (N + 1)]
000: reserved
001: PLLQ = 2
010: PLLQ = 3
011: PLLQ = 4
100: PLLQ = 5
101: PLLQ = 6
110: PLLQ = 7
111: PLLQ = 8

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 24 **PLLQEN**: Main PLL PLLQCLK output enable
Set and reset by software to enable the PLLQCLK output of the main PLL (used as system clock).
In order to save power, when the PLLQCLK output of the PLL is not used, the value of PLLQEN should be 0.
0: PLLQCLK output disable
1: PLLQCLK output enable

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **PLL[4:0]**: Main PLL division factor for PLLPCLK
Set and cleared by software to control the frequency of the main PLL output clock PLLPCLK. This output can be selected for SAI1 and ADC. These bits can be written only if PLL is disabled.

PLLPCLK output clock frequency = VCO frequency / PLLP = 2, 3, 4,... or 32 [VCO frequency / (N + 1)]
0000: reserved
00001: PLLP = 2
00010: PLLP = 3
00011: PLLP = 4
00100: PLLP = 5
...
11111: PLLP = 32

The software has to set these bits so that 64 MHz is not exceeded on this domain.

Bit 16 **PLL[4:0]**: Main PLL PLLPCLK output enable
Set and reset by software to enable the PLLPCLK output of the main PLL.
In order to save power, when the PLLPCLK output of the PLL is not used, the value of PLLPEN should be 0.
0: PLLPCLK output disable
1: PLLPCLK output enable

Bit 15 Reserved, must be kept at reset value.
Bits 14:8 **PLLN[6:0]**: Main PLL multiplication factor for VCO
Set and cleared by software to control the multiplication factor of the VCO. These bits can be written only when the PLL is disabled.
VCO output frequency = VCO input frequency x PLLN with $6 \leq PLLN \leq 127$
- 0000000: PLLN = 0 reserved (must not be used)
- 0000001: PLLN = 1 reserved (must not be used)
- ...
- 0000101: PLLN = 5 reserved (must not be used)
- 0000110: PLLN = 6
- 1111111: PLLN = 127

The software has to set these bits to ensure that the VCO output frequency is between 96 and 344 MHz.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PLLM[2:0]**: Division factor for the main PLL and audio PLLSAI1 input clock
Set and cleared by software to divide the PLL and PLLSAI1 input clock before the VCO. These bits can be written only when all PLLs are disabled.
VCO input frequency = PLL input clock frequency / PLLM with $1 \leq PLLM \leq 8$
- 000: PLLM = 1
- 001: PLLM = 2
- 010: PLLM = 3
- 011: PLLM = 4
- 100: PLLM = 5
- 101: PLLM = 6
- 110: PLLM = 7
- 111: PLLM = 8

The software has to set these bits to ensure that the VCO input frequency ranges from 2.66 to 16 MHz.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC[1:0]**: Main PLL and audio PLLSAI1 entry clock source
Set and cleared by software to select PLL and PLLSAI1 clock source. These bits can be written only when PLL and PLLSAI1 are disabled.
In order to save power, when no PLL is used, the value of PLLSRC should be 00.
- 00: No clock sent to PLL and PLLSAI1
- 01: MSI clock selected as PLL and PLLSAI1 clock entry
- 10: HSI16 clock selected as PLL and PLLSAI1 clock entry
- 11: HSE clock selected as PLL and PLLSAI1 clock entry
8.4.5 RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR)

Address offset: 0x010
Reset value: 0x2204 0100
Access: No wait state, word, half-word and byte access

This register is used to configure the PLLSAI1 clock outputs according to the formulas:

\[
\begin{align*}
  f(\text{VCOSAI clock}) &= f(\text{PLLSAI1 clock input}) \times (\text{PLLN} / \text{PLLM}) \\
  f(\text{PLLSAI1}_P) &= f(\text{VCOSAI clock}) / \text{PLLP} \\
  f(\text{PLLSAI1}_Q) &= f(\text{VCOSAI clock}) / \text{PLLQ} \\
  f(\text{PLLSAI1}_R) &= f(\text{VCOSAI clock}) / \text{PLLR}
\end{align*}
\]

<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 31:29 **PLLR[2:0]**: Audio PLLSAI1 division factor for PLLSAI1RCLK

Set and cleared by software to control the frequency of the audio PLLSAI1 output clock PLLSAI1RCLK. This output can be selected as system clock. These bits can be written only if PLLSAI1 is disabled.

PLLSAI1RCLK output clock frequency = VCO frequency / PLLR with PLLR = 2, 3, 4,... or 8

[\text{VCO frequency} / (\text{N} + 1)]

000: reserved
001: PLLR = 2
010: PLLR = 3
011: PLLR = 4
100: PLLR = 5
101: PLLR = 6
110: PLLR = 7
111: PLLR = 8

The software has to set these bits correctly not to exceed 64 MHz on this domain.

Bit 28 **PLLREN**: Audio PLLSAI1 PLLSAI1RCLK output enable

Set and reset by software to enable the PLLSAI1RCLK output of the audio PLLSAI1 (used as system clock).

In order to save power, when the PLLSAI1RCLK output of the PLLSAI1 is not used, the value of PLLREN should be 0.

0: PLLSAI1RCLK output disable
1: PLLSAI1RCLK output enable
Bits 27:25 **PLLQ[2:0]**: Audio PLLSAI1 division factor for PLLSAI1QCLK
Set and cleared by software to control the frequency of the audio PLLSAI1 output clock PLLSAI1QCLK. This output can be selected for USB and True RNG clock. These bits can be written only if PLLSAI1 is disabled.

PLLSAI1QCLK output clock frequency = \( \frac{VCO}{{PLLQ}} \) with PLLQ = 2, 3, 4, ... or 8

\[ \frac{VCO}{{(N + 1)}} \]

000: reserved
001: PLLQ = 2
010: PLLQ = 3
011: PLLQ = 4
100: PLLQ = 5
101: PLLQ = 6
110: PLLQ = 7
111: PLLQ = 8

The software has to set these bits correctly not to exceed 64 MHz on this domain.

Bit 24 **PLLQEN**: Audio PLLSAI1 PLLSAI1QCLK output enable
Set and reset by software to enable the PLLSAI1QCLK output of the audio PLLSAI1 (used as system clock).

In order to save power, when the PLLSAI1QCLK output of the PLLSAI1 is not used, the value of PLLQEN should be 0.

0: PLLSAI1QCLK output disable
1: PLLSAI1QCLK output enable

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **PLL[4:0]**: Audio PLLSAI1 division factor for PLLSAI1PCLK.
Set and cleared by software to control the frequency of the audio PLLSAI1 output clock PLLSAI1PCLK. This output can be selected for SAI1 and ADC. These bits can be written only if PLLSAI1 is disabled.

PLLSAI1PCLK output clock frequency = \( \frac{VCO}{{PLLP}} \) with PLLP = 2, 3, 4, ... or 32

\[ \frac{VCO}{{(N + 1)}} \]

00000: reserved
00001: PLLP = 2
00010: PLLP = 3
00011: PLLP = 4
00100: PLLP = 5
... 11111: PLLP = 32

Caution: The software has to set these bits correctly not to exceed 64 MHz on this domain.

Bit 16 **PLLPEN**: Audio PLLSAI1 PLLSAI1PCLK output enable
Set and reset by software to enable the PLLSAI1PCLK output of the audio PLLSAI1.
In order to save power, when the PLLSAI1PCLK output of the PLL is not used, the value of PLLPEN should be 0.

0: PLLSAI1PCLK output disable
1: PLLSAI1PCLK output enable

Bit 15 Reserved, must be kept at reset value.
Bits 14:8 **PLLN[6:0]**: Audio PLLSAI1 multiplication factor for VCO

Set and cleared by software to control the multiplication factor of the VCO. These bits can be written only when the PLLSAI1 is disabled.

VCO output frequency = VCO input frequency x PLLN with $4 \leq \text{PLLN} \leq 86$

0000000: PLLN = 0 wrong configuration

0000001: PLLN = 1 wrong configuration

... 0000011: PLLN = 3 wrong configuration

0000100: PLLN = 4

0000101: PLLN = 5

... 1010101: PLLN = 85

1010110: PLLN = 86

1010111: PLLN = 87 wrong configuration

... 1111111: PLLN = 127 wrong configuration

**Caution:** The software has to set correctly these bits to ensure that the VCO output frequency is between 64 and 344 MHz.

Bits 7:0 Reserved, must be kept at reset value.

### 8.4.6 RCC clock interrupt enable register (RCC_CIER)

Address offset: 0x018

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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>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 31:12 Reserved, must be kept at reset value.

**Bit 11 LSI2RdyIE**: LSI2 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI2 oscillator stabilization.

0: LSI2 ready interrupt disabled

1: LSI2 ready interrupt enabled

**Bit 10 HSI48RdyIE**: HSI48 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the internal HSI48 oscillator.

0: HSI48 ready interrupt disabled

1: HSI48 ready interrupt enabled
Bit 9 **LSECSSIE**: LSE clock security system interrupt enable
- Set and cleared by software to enable/disable interrupt caused by the clock security system on LSE.
  - 0: Clock security interrupt caused by LSE clock failure disabled
  - 1: Clock security interrupt caused by LSE clock failure enabled

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **PLLSAI1RDYIE**: PLLSAI1 ready interrupt enable
- Set and cleared by software to enable/disable interrupt caused by PLLSAI1 lock.
  - 0: PLLSAI1 lock interrupt disabled
  - 1: PLLSAI1 lock interrupt enabled

Bit 5 **PLLRDYIE**: PLL ready interrupt enable
- Set and cleared by software to enable/disable interrupt caused by PLL lock.
  - 0: PLL lock interrupt disabled
  - 1: PLL lock interrupt enabled

Bit 4 **HSERDYIE**: HSE ready interrupt enable
- Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.
  - 0: HSE ready interrupt disabled
  - 1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI16 ready interrupt enable
- Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization.
  - 0: HSI16 ready interrupt disabled
  - 1: HSI16 ready interrupt enabled

Bit 2 **MSIRDYIE**: MSI ready interrupt enable
- Set and cleared by software to enable/disable interrupt caused by the MSI oscillator stabilization.
  - 0: MSI ready interrupt disabled
  - 1: MSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
- Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
  - 0: LSE ready interrupt disabled
  - 1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI1 ready interrupt enable
- Set and cleared by software to enable/disable interrupt caused by the LSI1 oscillator stabilization.
  - 0: LSI1 ready interrupt disabled
  - 1: LSI1 ready interrupt enabled

8.4.7 **RCC clock interrupt flag register (RCC_CIFR)**

Address offset: 0x01C
Reset value: 0x0000 0000
Access: No wait state, word, half-word and byte access
**Reset and clock control (RCC)**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td><strong>LSI2RDYF</strong>: LSI2 ready interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware when the LSI2 clock becomes</td>
</tr>
<tr>
<td></td>
<td>stable and LSI2RDYDIE is set.</td>
</tr>
<tr>
<td></td>
<td>Cleared by software setting the LSI2RDYC bit.</td>
</tr>
<tr>
<td></td>
<td>0: No clock ready interrupt caused by the LSI2</td>
</tr>
<tr>
<td></td>
<td>oscillator</td>
</tr>
<tr>
<td></td>
<td>1: Clock ready interrupt caused by the LSI2</td>
</tr>
<tr>
<td></td>
<td>oscillator</td>
</tr>
<tr>
<td>10</td>
<td><strong>HSI48RDYF</strong>: HSI48 ready interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware when the HSI48 clock becomes</td>
</tr>
<tr>
<td></td>
<td>stable and HSI48RDYDIE is set in a response</td>
</tr>
<tr>
<td></td>
<td>to setting the HSI48ON (refer to RCC clock</td>
</tr>
<tr>
<td></td>
<td>recovery RC register (RCC_CRRCR)).</td>
</tr>
<tr>
<td></td>
<td>Cleared by software setting the HSI48RDYC bit.</td>
</tr>
<tr>
<td></td>
<td>0: No clock ready interrupt caused by the HSI48</td>
</tr>
<tr>
<td></td>
<td>oscillator</td>
</tr>
<tr>
<td></td>
<td>1: Clock ready interrupt caused by the HSI48</td>
</tr>
<tr>
<td></td>
<td>oscillator</td>
</tr>
<tr>
<td>9</td>
<td><strong>LSECSSF</strong>: LSE clock security system</td>
</tr>
<tr>
<td></td>
<td>interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware when a failure is detected in</td>
</tr>
<tr>
<td></td>
<td>the LSE oscillator.</td>
</tr>
<tr>
<td></td>
<td>Cleared by software setting the LSECSSSC bit.</td>
</tr>
<tr>
<td></td>
<td>0: No clock security interrupt caused by LSE</td>
</tr>
<tr>
<td></td>
<td>clock failure</td>
</tr>
<tr>
<td></td>
<td>1: Clock security interrupt caused by LSE</td>
</tr>
<tr>
<td></td>
<td>clock failure</td>
</tr>
<tr>
<td>8</td>
<td><strong>CSSF</strong>: HSE clock security system interrupt</td>
</tr>
<tr>
<td></td>
<td>flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware when a failure is detected in</td>
</tr>
<tr>
<td></td>
<td>the HSE oscillator.</td>
</tr>
<tr>
<td></td>
<td>Cleared by software setting the CSSC bit.</td>
</tr>
<tr>
<td></td>
<td>0: No clock security interrupt caused by HSE</td>
</tr>
<tr>
<td></td>
<td>clock failure</td>
</tr>
<tr>
<td></td>
<td>1: Clock security interrupt caused by HSE</td>
</tr>
<tr>
<td></td>
<td>clock failure</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td><strong>PLLSA1RDYF</strong>: PLLSA1 ready interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware when the PLLSA1 locks and</td>
</tr>
<tr>
<td></td>
<td>PLLSA1RDYDIE is set.</td>
</tr>
<tr>
<td></td>
<td>Cleared by software setting the PLLSA1RDYC bit.</td>
</tr>
<tr>
<td></td>
<td>0: No clock ready interrupt caused by PLLSA1</td>
</tr>
<tr>
<td></td>
<td>lock</td>
</tr>
<tr>
<td></td>
<td>1: Clock ready interrupt caused by PLLSA1 lock</td>
</tr>
<tr>
<td>5</td>
<td><strong>PLLRDYF</strong>: PLL ready interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware when the PLL locks and PLLRDYD</td>
</tr>
<tr>
<td></td>
<td>DIE is set.</td>
</tr>
<tr>
<td></td>
<td>Cleared by software setting the PLLRDYC bit.</td>
</tr>
<tr>
<td></td>
<td>0: No clock ready interrupt caused by PLL lock</td>
</tr>
<tr>
<td></td>
<td>1: Clock ready interrupt caused by PLL lock</td>
</tr>
<tr>
<td>4</td>
<td><strong>HSERDYF</strong>: HSE ready interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware when the HSE clock becomes</td>
</tr>
<tr>
<td></td>
<td>stable and HSERDYDIE is set.</td>
</tr>
<tr>
<td></td>
<td>Cleared by software setting the HSERDYC bit.</td>
</tr>
<tr>
<td></td>
<td>0: No clock ready interrupt caused by the HSE</td>
</tr>
<tr>
<td></td>
<td>oscillator</td>
</tr>
<tr>
<td></td>
<td>1: Clock ready interrupt caused by the HSE</td>
</tr>
<tr>
<td></td>
<td>oscillator</td>
</tr>
</tbody>
</table>
8.4.8 RCC clock interrupt clear register (RCC_CICR)

Address offset: 0x020

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>

<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>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 31:12 Reserved, must be kept at reset value.

Bit 11 **LSI2RDYC**: LSI2 ready interrupt clear
This bit is set by software to clear the LSI2RDYF flag.
0: No effect
1: LSI2RDYF cleared

Bit 10 **HSI48RDYC**: HSI48 oscillator ready interrupt clear
This bit is set by software to clear the HSI48RDYF flag.
0: No effect
1: Clear the HSI48RDYC flag
8.4.9  RCC SMPS step-down converter control register (RCC_SMPSCR)

Address offset: 0x024

Reset value: 0x0000 0301 (after POR reset), 0x0000 0300 (after wakeup from Standby)

Access: No wait state, word, half-word and byte access

Bit 9  **LSECSSC**: LSE clock security system interrupt clear
   This bit is set by software to clear the LSECSSF flag.
   0: No effect
   1: Clear LSECSSF flag

Bit 8  **CSSC**: HSE clock security system interrupt clear
   This bit is set by software to clear the HSE CSSF flag.
   0: No effect
   1: Clear HSE CSSF flag

Bit 7  Reserved, must be kept at reset value.

Bit 6  **PLLSAI1RDYC**: PLLSAI1 ready interrupt clear
   This bit is set by software to clear the PLLSAI1RDYF flag.
   0: No effect
   1: Clear PLLSAI1RDYF flag

Bit 5  **PLLRDYC**: PLL ready interrupt clear
   This bit is set by software to clear the PLLRDYF flag.
   0: No effect
   1: Clear PLLRDYF flag

Bit 4  **HSERDYC**: HSE ready interrupt clear
   This bit is set by software to clear the HSERDYF flag.
   0: No effect
   1: Clear HSERDYF flag

Bit 3  **HSIRDYC**: HSI16 ready interrupt clear
   This bit is set software to clear the HSIRDYF flag.
   0: No effect
   1: Clear HSIRDYF flag

Bit 2  **MSIRDYC**: MSI ready interrupt clear
   This bit is set by software to clear the MSIRDYF flag.
   0: No effect
   1: MSIRDYF cleared

Bit 1  **LSEIRDYC**: LSE ready interrupt clear
   This bit is set by software to clear the LSEIRDYF flag.
   0: No effect
   1: LSEIRDYF cleared

Bit 0  **LSI1RDYC**: LSI1 ready interrupt clear
   This bit is set by software to clear the LSI1RDYF flag.
   0: No effect
   1: LSI1RDYF cleared
Note: On a POR reset or NRST pin reset or when waking up from Shutdown the SMPSSEL is forced by hardware to select MSI (value b01). When waking up from Standby the SMPSSEL is forced by hardware to select HSI (value b00). When waking up from Stop modes the SMPSSEL is forced by hardware to select the MSI or HSI clock as defined in RCC_CFGR.STOPWUCK.

8.4.10 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x028

Reset value: 0x00000 0000

Access: No wait state, word, half-word and byte access
### Reset and clock control (RCC)

#### 8.4.11 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x02C
Reset value: 0x00000 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>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>
</tr>
</tbody>
</table>

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

- **Bit 16**: **TSCRST**: Touch sensing controller reset (STM32WB55xx only)
  - Set and cleared by software.
  - 0: No effect
  - 1: Reset TSC
  - Note that this bit is reserved on STM32WB35xx.

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 11:3 Reserved, must be kept at reset value.

- **Bit 2**: **DMAMUX1RST**: DMAMUX reset
  - Set and cleared by software.
  - 0: No effect
  - 1: Reset DMAMUX1

- **Bit 1**: **DMA2RST**: DMA2 reset
  - Set and cleared by software.
  - 0: No effect
  - 1: Reset DMA2

- **Bit 0**: **DMA1RST**: DMA1 reset
  - Set and cleared by software.
  - 0: No effect
  - 1: Reset DMA1
Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AES1RST**: AES1 hardware accelerator reset
Set and cleared by software.
0: No effect
1: Reset AES1

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ACRST**: ADC reset
Set and cleared by software.
0: No effect
1: Reset ADC interface

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST**: IO port H reset
Set and cleared by software.
0: No effect
1: Reset IO port H

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOERST**: IO port E reset
Set and cleared by software.
0: No effect
1: Reset IO port E

Bit 3 **GPIODRST**: IO port D reset (STM32WB55xx only)
Set and cleared by software.
0: No effect
1: Reset IO port D
Note that this bit is reserved on STM32WB35xx.

Bit 2 **GPIOCRST**: IO port C reset
Set and cleared by software.
0: No effect
1: Reset IO port C

Bit 1 **GPIOBRST**: IO port B reset
Set and cleared by software.
0: No effect
1: Reset IO port B

Bit 0 **GPIOARST**: IO port A reset
Set and cleared by software.
0: No effect
1: Reset IO port A

8.4.12 **RCC AHB3 and AHB4 peripheral reset register (RCC_AHB3RSTR)**
Address offset: 0x030
Reset value: 0x00000 0000
Access: No wait state, word, half-word and byte access
### Reset and clock control (RCC)

**8.4.13 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)**

Address offset: 0x038

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset: 0x038</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</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>
<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>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></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>
<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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:26  Reserved, must be kept at reset value.

**Bit 25 FLASHRST:** Flash interface reset
- This bit can only be set when the Flash memory is in power-down. Set and cleared by software.
- 0: No effect
- 1: Reset Flash interface.

Bits 24:21  Reserved, must be kept at reset value.

**Bit 20 IPCCRST:** IPCC interface reset
- Set and cleared by software.
- 0: No effect
- 1: Reset IPCC

**Bit 19 HSEMRST:** HSEM reset
- Set and cleared by software.
- 0: No effect
- 1: Reset HSEM

**Bit 18 RNGRST:** True RNG reset
- Set and cleared by software.
- 0: No effect
- 1: Reset true RNG

**Bit 17 AES2RST:** AES2 hardware accelerator reset
- Set and cleared by software.
- 0: No effect
- 1: Reset AES2

**Bit 16 PKARST:** PKA hardware accelerator reset
- Set and cleared by software.
- 0: No effect
- 1: Reset PKA

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

**Bit 8 QUADSPIRST:** Quad SPI memory interface reset
- Set and cleared by software.
- 0: No effect
- 1: Reset QUADSPI

Bits 7:0  Reserved, must be kept at reset value.
Access: No wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30-27</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 20-15</th>
<th>Bit 14</th>
<th>Bit 13-10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LPTIM1RST</strong>: Low power timer 1 reset</td>
<td>Reserved, must be kept at reset value.</td>
<td><strong>USBRST</strong>: USB FS reset</td>
<td>Reserved, must be kept at reset value</td>
<td><strong>CRSRST</strong>: CRS reset</td>
<td>Reserved, must be kept at reset value</td>
<td><strong>I2C3RST</strong>: I2C3 reset</td>
<td>Reserved, must be kept at reset value</td>
<td><strong>I2C1RST</strong>: I2C1 reset</td>
<td>Reserved, must be kept at reset value.</td>
<td><strong>SPI2RST</strong>: SPI2 reset (STM32WB55xx only)</td>
</tr>
<tr>
<td>Set and cleared by software.</td>
<td>0: No effect</td>
<td>1: Reset USB FS</td>
<td>Set and cleared by software.</td>
<td>0: No effect</td>
<td>1: Reset the CRS</td>
<td>Set and reset by software.</td>
<td>0: No effect</td>
<td>1: Resets I2C3</td>
<td>Set and cleared by software.</td>
<td>Set and cleared by software.</td>
</tr>
<tr>
<td>0: No effect</td>
<td>1: Reset LPTIM1</td>
<td>0: No effect</td>
<td>1: Reset the USB FS</td>
<td>0: No effect</td>
<td>1: Reset the CRS</td>
<td>0: No effect</td>
<td>1: Resets I2C3</td>
<td>0: No effect</td>
<td>1: Reset I2C1</td>
<td></td>
</tr>
</tbody>
</table>

Note that this bit is reserved on STM32WB35xx.
Bit 9 **LCDRST**: LCD interface reset (STM32WB55xx only)
Set and cleared by software.
0: No effect
1: Reset LCD
Note that this bit is reserved on STM32WB35xx.

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **TIM2RST**: TIM2 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM2

### 8.4.14 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x03C
Reset value: 0x00000 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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 **LPTIM2RST**: Low-power timer 2 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM2

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPUART1RST**: Low-power UART 1 reset
Set and cleared by software.
0: No effect
1: Reset LPUART1

### 8.4.15 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x040
Reset value: 0x00000 0000
Access: No wait state, word, half-word and byte access
## Reset and clock control (RCC)

### RCC_APB3 peripheral reset register (RCC_APB3RSTR)

**Address offset:** 0x044  
**Reset value:** 0x00000000  
**Access:** No wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td><strong>SAI1RST:</strong> Serial audio interface 1 (SAI1) reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No effect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Reset SAI1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td><strong>TIM17RST:</strong> TIM17 timer reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No effect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Reset TIM17 timer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td><strong>TIM16RST:</strong> TIM16 timer reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No effect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Reset TIM16 timer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td><strong>USART1RST:</strong> USART1 reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No effect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Reset USART1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td><strong>SPI1RST:</strong> SPI1 reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No effect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Reset SPI1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td><strong>TIM1RST:</strong> TIM1 timer reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No effect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Reset TIM1 timer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 10:0 Reserved, must be kept at reset value.**

### 8.4.16 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x044  
Reset value: 0x00000000  
Access: No wait state, word, half-word and byte access
# Reset and clock control (RCC)

## 8.4.17 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

| Address offset: 0x048 | Reset value: 0x0000 0000 | Access: No wait state, word, half-word and byte access |

**Note:** When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

<table>
<thead>
<tr>
<th>Bit 31:17 Reserved</th>
<th>Bit 16 <strong>TSCEN:</strong> CPU1 Touch sensing controller clock enable (STM32WB55xx only)</th>
<th>Set and cleared by software.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15:13 Reserved</td>
<td>Bit 12 <strong>CRCEN:</strong> CPU1 CRC clock enable</td>
<td>Set and cleared by software.</td>
</tr>
<tr>
<td>Bit 11:3 Reserved</td>
<td>Bit 2 <strong>DMAMUX1:</strong> CPU1 DMAMUX1 clock enable</td>
<td>Set and cleared by software.</td>
</tr>
</tbody>
</table>

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

Bit 0 **RFRST:** Radio system BLE and 802.15.4 reset.

- Set and cleared by software.
- 0: No effect
- 1: Reset radio system BLE and 802.15.4. The reset status of the radio system can be obtained from RFRSTS in **RCC control/status register (RCC_CSR)**.

<table>
<thead>
<tr>
<th>Bit 0 <strong>RFRST:</strong></th>
<th>Radio system BLE and 802.15.4 reset</th>
<th>Set and cleared by software.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No effect</td>
<td>0: No effect</td>
</tr>
<tr>
<td>1</td>
<td>Reset radio system BLE and 802.15.4</td>
<td>1: Reset radio system BLE and 802.15.4. The reset status of the radio system can be obtained from RFRSTS in <strong>RCC control/status register (RCC_CSR)</strong>.</td>
</tr>
</tbody>
</table>

**Notes:**

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

**Bit 0 RFRST:** Radio system BLE and 802.15.4 reset.

- Set and cleared by software.
- 0: No effect
- 1: Reset radio system BLE and 802.15.4. The reset status of the radio system can be obtained from RFRSTS in **RCC control/status register (RCC_CSR)**.

**8.4.17 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)**

Address offset: 0x048

Reset value: 0x0000 0000

Access: No wait state, word, half-word and byte access

**Note:** When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

<table>
<thead>
<tr>
<th>Bit 16 <strong>TSCEN:</strong></th>
<th>CPU1 Touch sensing controller clock enable (STM32WB55xx only)</th>
<th>Set and cleared by software.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TSC clock disable for CPU1</td>
<td>0: TSC clock disable for CPU1</td>
</tr>
<tr>
<td>1</td>
<td>TSC clock enable for CPU1</td>
<td>1: TSC clock enable for CPU1</td>
</tr>
</tbody>
</table>

**Note:** This bit is reserved on STM32WB35xx.

**Bits 15:13:** Reserved, must be kept at reset value.

**Bit 12 **CRCEN:** CPU1 CRC clock enable | Set and cleared by software. |

- 0: CRC clock disable for CPU1
- 1: CRC clock enable for CPU1

**Bits 11:3:** Reserved, must be kept at reset value.

**Bit 2 **DMAMUX1:** CPU1 DMAMUX1 clock enable | Set and cleared by software. |

- 0: DMAMUX1 clock disable for CPU1
- 1: DMAMUX1 clock enable for CPU1
8.4.18 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x04C
Reset value: 0x0000 0000
Access: No wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>AES1EN: CPU1 AES1 accelerator clock enable</td>
<td>Set and cleared by software.</td>
<td>0: AES clock disabled for CPU1 1: AES clock enabled for CPU1</td>
</tr>
<tr>
<td>15:14</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>ADCEN: CPU1 ADC clocks enable</td>
<td>Set and cleared by software.</td>
<td>0: ADC bus and kernel clocks disabled for CPU1 1: ADC bus and kernel clocks enabled for CPU1</td>
</tr>
<tr>
<td>12:8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GPIOHEN: CPU1 IO port H clock enable</td>
<td>Set and cleared by software.</td>
<td>0: IO port H clock disabled for CPU1 1: IO port H clock enabled for CPU1</td>
</tr>
<tr>
<td>6:5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GPIOEEN: CPU1 IO port E clock enable</td>
<td>Set and cleared by software.</td>
<td>0: IO port E clock disabled for CPU1 1: IO port E clock enabled for CPU1</td>
</tr>
</tbody>
</table>
Bit 3 **GPIODEN**: CPU1 IO port D clock enable (STM32WB55xx only)
   - Set and cleared by software.
   - 0: IO port D clock disabled for CPU1
   - 1: IO port D clock enabled for CPU1
   - Note that this bit is reserved on STM32WB35xx.

Bit 2 **GPIOCEN**: CPU1 IO port C clock enable
   - Set and cleared by software.
   - 0: IO port C clock disabled for CPU1
   - 1: IO port C clock enabled for CPU1

Bit 1 **GPIOBEN**: CPU1 IO port B clock enable
   - Set and cleared by software.
   - 0: IO port B clock disabled for CPU1
   - 1: IO port B clock enabled for CPU1

Bit 0 **GPIOAEN**: CPU1 IO port A clock enable
   - Set and cleared by software.
   - 0: IO port A clock disabled for CPU1
   - 1: IO port A clock enabled for CPU1

### 8.4.19 RCC AHB3 and AHB4 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x050

Reset value: 0x0208 0000

Access: No wait state, word, half-word and byte access

**Note**: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

<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>
</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>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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>
<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>
<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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHEN** CPU1 Flash memory interface clock enable
   - This bit can only be cleared when the Flash memory is in Power down. Set and cleared by software.
   - 0: Flash interface clock disable for CPU1
   - 1: Flash interface clock enable for CPU1

Bits 24:21 Reserved, must be kept at reset value.

Bit 20 **IPCCEN** CPU1 IPCC interface clock enable
   - Set and cleared by software.
   - 0: IPCC clock disable for CPU1
   - 1: IPCC clock enable for CPU1
Bit 19 **HSEMEN** CPU1 HSEM clock enable
Set and cleared by software.
0: HSEM clock disable for CPU1
1: HSEM clock enable for CPU1

Bit 18 **RNGEN** CPU1 true RNG clocks enable
Set and cleared by software.
0: True RNG bus and kernel clocks disable for CPU1
1: True RNG bus and kernel clocks enable for CPU1

Bit 17 **AES2EN** CPU1 AES2 accelerator clock enable
Set and cleared by software.
0: AES2 clock disable for CPU1
1: AES2 clock enable for CPU1

Bit 16 **PKAEN** CPU1 PKA accelerator clock enable
Set and cleared by software.
0: PKA clock disable for CPU1
1: PKA clock enable for CPU1

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

Bit 8 **QUADSPIEN**: CPU1 Quad SPI memory interface clock enable
Set and cleared by software.
0: QUADSPI clock disable for CPU1
1: QUADSPI clock enable for CPU1

Bits 7:0 Reserved, must be kept at reset value.

### 8.4.20 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address: 0x058
Reset value: 0x0000 0400
Access: No wait state, word, half-word and byte access

**Note:** When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>LPTIM1EN</strong>: CPU1 Low power timer 1 clocks enable</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td><strong>SpI2EN</strong></td>
<td>rs</td>
</tr>
<tr>
<td>29</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td><strong>USBEN</strong></td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td><strong>Res</strong></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **LPTIM1EN**: CPU1 Low power timer 1 clocks enable
Set and cleared by software.
0: LPTIM1 bus and kernel clocks disabled for CPU1
1: LPTIM1 bus and kernel clocks enabled for CPU1

Bits 30:27 Reserved, must be kept at reset value.
Bit 26 **USBEN**: CPU1 USB FS clocks enable
Set and cleared by software
0: USB FS bus and kernel clocks disabled for CPU1
1: USB FS bus and kernel clocks enabled for CPU1

Bits 25 Reserved, must be kept at reset value.

Bit 24 **CRSEN**: CPU1 CRS clock enable
Set and cleared by software.
0: CRS clock disabled for CPU1
1: CRS clock enabled for CPU1

Bit 23 **I2C3EN**: CPU1 I2C3 clocks enable
Set and cleared by software.
0: I2C3 bus and kernel clocks disabled for CPU1
1: I2C3 bus and kernel clocks enabled for CPU1

Bits 22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: CPU1 I2C1 clocks enable
Set and cleared by software.
0: I2C1 bus and kernel clocks disabled for CPU1
1: I2C1 bus and kernel clocks enabled for CPU1

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI2EN**: CPU1 SPI2 clock enable (STM32WB55xx only)
Set and cleared by software.
0: SPI2 clock disabled for CPU1
1: SPI2 clock enabled for CPU1
Note that this bit is reserved on STM32WB35xx.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: CPU1 Window watchdog clock enable
Set by software to enable the window watchdog clock. Reset by hardware system reset.
This bit can also be set by hardware if the WWDG_SW option bit is reset.
0: Window watchdog clock disabled for CPU1
1: Window watchdog clock enabled for CPU1

Bit 10 **RTCAPBEN**: CPU1 RTC APB clock enable
Set and cleared by software
0: RTC APB clock disabled for CPU1
1: RTC APB clock enabled for CPU1

Bit 9 **LCDEN**: CPU1 LCD clock enable (STM32WB55xx only)
Set and cleared by software.
0: LCD clock disabled for CPU1
1: LCD clock enabled for CPU1
Note that this bit is reserved on STM32WB35xx.

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **TIM2EN**: CPU1 TIM2 timer clock enable
Set and cleared by software.
0: TIM2 clock disabled for CPU1
1: TIM2 clock enabled for CPU1
8.4.21 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x05C
Reset value: 0x00000 0000
Access: No wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

| 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

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

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

Bit 5 LPTIM2EN CPU1 Low power timer 2 clocks enable
Set and cleared by software.
0: LPTIM2 bus and kernel clocks disabled for CPU1
1: LPTIM2 bus and kernel clocks enabled for CPU1

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 LPUART1EN CPU1 Low power UART 1 clocks enable
Set and cleared by software.
0: LPUART1 bus and kernel clocks disabled for CPU1
1: LPUART1 bus and kernel clocks enabled for CPU1

8.4.22 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x060
Reset value: 0x00000 0000
Access: word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU1 is not supported.

| 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

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

Bits 31:22 Reserved, must be kept at reset value.
Bit 21 **SAI1EN**: CPU1 SAI1 clocks enable
Set and cleared by software.
0: SAI1 bus and kernel clocks disabled for CPU1
1: SAI1 bus and kernel clocks enabled for CPU1

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: CPU1 TIM17 timer clock enable
Set and cleared by software.
0: TIM17 timer clock disabled for CPU1
1: TIM17 timer clock enabled for CPU1

Bit 17 **TIM16EN**: CPU1 TIM16 timer clock enable
Set and cleared by software.
0: TIM16 timer clock disabled for CPU1
1: TIM16 timer clock enabled for CPU1

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: CPU1 USART1 clocks enable
Set and cleared by software.
0: USART1 bus and kernel clocks disabled for CPU1
1: USART1 bus and kernel clocks enabled for CPU1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: CPU1 SPI1 clock enable
Set and cleared by software.
0: SPI1 clock disabled for CPU1
1: SPI1 clock enabled for CPU1

Bit 11 **TIM1EN**: CPU1 TIM1 timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled for CPU1
1: TIM1P timer clock enabled for CPU1

Bits 10:0 Reserved, must be kept at reset value.

### 8.4.23 RCC AHB1 peripheral clocks enable in Sleep modes register (RCC_AHB1SMENR)

Address offset: 0x068
Reset value: 0x0001 1207
Access: No wait state, word, half-word and byte access
Bits 31:17 Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td><strong>TSCSMEN:</strong> Touch sensing controller clock enable during CPU1 CSleep mode (STM32WB55xx only)</td>
<td>Set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: TSC clock disabled by the clock gating during CPU1 Csleep and CStop modes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: TSC clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note that this bit is reserved on STM32WB35xx.</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

| Bit 12 | **CRCSMEN:** CRC clock enable during CPU1 CSleep mode                                                | Set and cleared by software.                        |
|        | 0: CRC clock disabled by the clock gating during CPU1 Csleep and CStop modes                         |              |
|        | 1: CRC clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode    |              |

Bits 11:10 Reserved, must be kept at reset value.

| Bit 9  | **SRAM1SMEN:** SRAM1 interface clock enable during CPU1 CSleep mode                                  | Set and cleared by software.                        |
|        | 0: SRAM1 interface clock disabled by the clock gating during CPU1 Csleep and CStop modes             |              |
|        | 1: SRAM1 interface clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode |              |

Bits 8:3 Reserved, must be kept at reset value.

| Bit 2  | **DMAMUX1SMEN:** DMAMUX1 clock enable during CPU1 CSleep mode                                        | Set and cleared by software during Sleep mode.       |
|        | 0: DMAMUX1 clock disabled by the clock gating during CPU1 Csleep and CStop modes                      |              |
|        | 1: DMAMUX1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode |              |

| Bit 1  | **DMA2SMEN:** DMA2 clock enable during CPU1 CSleep mode                                               | Set and cleared by software during Sleep mode.       |
|        | 0: DMA2 clock disabled by the clock gating during CPU1 Csleep and CStop modes                         |              |
|        | 1: DMA2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode    |              |

| Bit 0  | **DMA1SMEN:** DMA1 clock enable during CPU1 CSleep mode                                               | Set and cleared by software.                        |
|        | 0: DMA1 clock disabled by the clock gating during CPU1 Csleep and CStop modes                         |              |
|        | 1: DMA1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode    |              |

**8.4.24 RCC AHB2 peripheral clocks enable in Sleep modes register (RCC_AHB2SMENR)**

Address offset: 0x06C
Reset value: 0x0001 209F
Access: No wait state, word, half-word and byte access
## Reset and clock control (RCC)

<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></td>
<td></td>
<td></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>
<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 31:17  Reserved, must be kept at reset value.

- **Bit 16 AES1SMEN**: AES1 accelerator clock enable during CPU1 CSleep mode
  - Set and cleared by software.
  - 0: AES1 clock disabled by the clock gating during CPU1 Csleep and CStop modes
  - 1: AES1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bits 15:14  Reserved, must be kept at reset value.

- **Bit 13 ADCSMEN**: ADC clock enable during CPU1 Csleep and CStop modes
  - Set and cleared by software.
  - 0: ADC bus clock disabled by the clock gating during CPU1 Csleep and CStop modes.
  - 1: ADC bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 12:8  Reserved, must be kept at reset value.

- **Bit 7 GPIOHSMEN**: IO port H clock enable during CPU1 CSleep mode
  - Set and cleared by software.
  - 0: IO port H clock disabled by the clock gating during CPU1 Csleep and CStop modes
  - 1: IO port H clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 6:5  Reserved, must be kept at reset value.

- **Bit 4 GPIOESMEN**: IO port E clock enable during CPU1 CSleep mode
  - Set and cleared by software.
  - 0: IO port E clock disabled by the clock gating during CPU1 Csleep and CStop modes
  - 1: IO port E clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

- **Bit 3 GPIODSMEN**: IO port D clock enable during CPU1 CSleep mode (STM32WB55xx only)
  - Set and cleared by software.
  - 0: IO port D clock disabled by the clock gating during CPU1 Csleep and CStop modes
  - 1: IO port D clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
  - Note that this bit is reserved on STM32WB35xx.

- **Bit 2 GPIOCSMEN**: IO port C clock enable during CPU1 CSleep mode
  - Set and cleared by software.
  - 0: IO port C clock disabled by the clock gating during CPU1 Csleep and CStop modes
  - 1: IO port C clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode
8.4.25 **RCC AHB3 and AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR)**

Address offset: 0x070  
Reset value: 0x0307 0100  
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, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
| 25  | FLASHSMEN: Flash memory interface clock enable during CPU1 CSleep mode      | 0: Flash interface clock disabled by the clock gating during CPU1 Csleep and CStop modes  
                                            | 1: Flash interface clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode  |
| 24  | SRAM2SMEN: SRAM2a and SRAM2b memory interface clock enable during CPU1 CSleep mode | 0: SRAM2 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
                                            | 1: SRAM2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode  |
| 23  | RNGSMEN: True RNG clock enable during CPU1 Csleep and CStop modes           | 0: True RNG bus clock disabled by the clock gating during CPU1 Csleep and CStop modes  
                                            | 1: True RNG bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode  |
| 22  | AES2SMEN                                                                    |                                                                     |
| 21  | PKASMEN                                                                     |                                                                     |
| 20  | Reserved                                                                    |                                                                     |
| 19  | Reserved                                                                    |                                                                     |
| 18  | Reserved                                                                    |                                                                     |
| 17  | Reserved                                                                    |                                                                     |
| 16  | Reserved                                                                    |                                                                     |

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHSMEN:** Flash memory interface clock enable during CPU1 CSleep mode  
Set and cleared by software.  
0: Flash interface clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: Flash interface clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 24 **SRAM2SMEN:** SRAM2a and SRAM2b memory interface clock enable during CPU1 CSleep mode  
Set and cleared by software.  
0: SRAM2 clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: SRAM2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **RNGSMEN:** True RNG clock enable during CPU1 Csleep and CStop modes  
Set and cleared by software.  
0: True RNG bus clock disabled by the clock gating during CPU1 Csleep and CStop modes  
1: True RNG bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.
Bit 17 **AES2MEN**: AES2 accelerator clock enable during CPU1 CSleep mode
Set and cleared by software.
0: AES2 clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: AES2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode

Bit 16 **PKASMEN**: PKA accelerator clock enable during CPU1 CSleep mode
Set and cleared by software.
0: PKA clock disabled by the clock gating during CPU1 CSleep and CStop modes
1: PKA clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

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

Bit 8 **QUADSPISMEN**: Quad SPI memory interface clock enable during CPU1 CSleep mode
Set and cleared by software.
0: QUADSPI clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: QUADSPI clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 7:0 Reserved, must be kept at reset value.

### 8.4.26 RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC_APB1SMENR1)

Address: 0x078
Reset value: 0x85A0 4E01
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>rw</td>
<td>rw</td>
<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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 **LPTIM1SMEN**: Low power timer 1 clock enable during CPU1 Csleep and CStop mode
Set and cleared by software.
0: LPTIM1 bus clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: LPTIM1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 30:27 Reserved, must be kept at reset value.

Bit 26 **USBSMEN**: USB FS clock enable during CPU1 Csleep and CStop modes
Set and cleared by software.
0: USB FS bus clock disabled by the clock gating during CPU1 Csleep and CStop modes
1: USB FS bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 25 Reserved, must be kept at reset value.
Bit 24 **CRSSMEN**: CRS clock enable during CPU1 CSleep mode  
Set and cleared by software.  
0: CRS clock disabled by the clock gating during CPU1 CSleep and CStop modes  
1: CRS clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 23 **I2C3SMEN**: I2C3 clock enable during CPU1 CSleep and CStop modes  
Set and cleared by software.  
0: I2C3 bus clock disabled by the clock gating during CPU1 CSleep and CStop modes  
1: I2C3 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN**: I2C1 clock enable during CPU1 CSleep and CStop modes  
Set and cleared by software.  
0: I2C1 bus clock disabled by the clock gating during CPU1 CSleep and CStop modes  
1: I2C1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI2SMEN**: SPI2 clock enable during CPU1 CSleep mode (STM32WB55xx only)  
Set and cleared by software.  
0: SPI2 clock disabled by the clock gating during CPU1 CSleep and CStop modes  
1: SPI2 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.  
Note that this bit is reserved on STM32WB35xx.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSSEN**: Window watchdog clocks enable during CPU1 CSleep mode  
Set and cleared by software. This bit is forced to ‘1’ by hardware when the hardware WWDG_SW option is reset.  
0: Window watchdog clock disabled by the clock gating during CPU1 CSleep and CStop modes  
1: Window watchdog clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 10 **RTCAPBSMEN**: RTC bus clock enable during CPU1 CSleep mode  
Set and cleared by software.  
0: RTC bus clock disabled during by the clock gating during CPU1 CSleep and CStop modes  
1: RTC bus clock enabled during by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 9 **LCDMEN**: LCD clock enable during CPU1 CSleep mode (STM32WB55xx only)  
Set and cleared by software.  
0: LCD clock disabled by the clock gating during CPU1 CSleep and CStop modes  
1: LCD clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.  
Note that this bit is reserved on STM32WB35xx.

Bits 8:1 Reserved, must be kept at reset value.
### 8.4.27 RCC APB1 peripheral clocks enable in Sleep mode register 2 (RCC_APB1SMENR2)

Address offset: 0x07C

Reset value: 0x0000 0021

Access: No wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit 31-6</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

| Bit 5 | LPTIM2SMEN | Low power timer 2 clock enable during CPU1 CSleep and CStop modes |
|       | Set and cleared by software. |
|       | 0: LPTIM2 bus clock disabled by the clock gating during CPU1 CSleep and CStop modes. |
|       | 1: LPTIM2 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode. |

| Bit 0 | LPUART1SMEN: Low power UART 1 clock enable during CPU1 CSleep and CStop modes. Set and cleared by software. |
|       | 0: LPUART1 bus clock disabled by the clock gating during CPU1 CSleep and CStop modes. |
|       | 1: LPUART1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode. |

### 8.4.28 RCC APB2 peripheral clocks enable in Sleep mode register (RCC_APB2SMENR)

Address: 0x080

Reset value: 0x0026 5800

Access: word, half-word and byte access
Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SMEN**: SAI1 clock enable during CPU1 Csleep and CStop modes
- Set and cleared by software.
  - 0: SAI1 bus clock disabled by the clock gating during CPU1 Csleep and CStop mode
  - 1: SAI1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM17 timer clock enable during CPU1 CSleep mode
- Set and cleared by software.
  - 0: TIM17 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode
  - 1: TIM17 timer clocks enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 17 **TIM16SMEN**: TIM16 timer clock enable during CPU1 CSleep mode
- Set and cleared by software.
  - 0: TIM16 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode
  - 1: TIM16 timer clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN**: USART1 clock enable during CPU1 Csleep and CStop modes.
- Set and cleared by software.
  - 0: USART1 bus clock disabled by the clock gating during CPU1 Csleep and CStop mode
  - 1: USART1 bus clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN**: SPI1 clock enable during CPU1 CSleep mode
- Set and cleared by software.
  - 0: SPI1 clock disabled by the clock gating during CPU1 Csleep and CStop mode
  - 1: SPI1 clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bit 11 **TIM1SMEN**: TIM1 timer clock enable during CPU1 CSleep mode
- Set and cleared by software.
  - 0: TIM1 timer clock disabled by the clock gating during CPU1 Csleep and CStop mode
  - 1: TIM1 timer clock enabled by the clock gating during CPU1 CSleep mode, disabled during CPU1 CStop mode.

Bits 10:0 Reserved, must be kept at reset value.
### 8.4.29 RCC peripherals independent clock configuration register (RCC_CCIPR)

Address: 0x088
Reset value: 0x0000 0000
Access: no wait states, word, half-word and byte access

<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>rw</td>
<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>
</tr>
<tr>
<td>Res</td>
<td>Res</td>
<td>Rs</td>
<td>Rs</td>
<td>Rs</td>
<td>Rs</td>
<td>Rs</td>
<td>Rs</td>
<td>Rs</td>
</tr>
</tbody>
</table>

#### Bits 31:30 RNGSEL[1:0]: RNG clock source selection
These bits are set and cleared by software to select the clock source used by the true RNG.
- **00**: Use clock as selected by CLK48SEL
- **01**: Select LSI clock
- **10**: Select LSE clock
- **11**: Reserved

#### Bits 29:28 ADCSEL[1:0]: ADC clock source selection
These bits are set and cleared by software to select the clock source used by the ADC interface.
- **00**: No clock selected
- **01**: PLLSAI1 "R" clock (PLLSAI1RCLK) selected as ADC clock
- **10**: PLL "P" clock (PLLCLK) selected as ADC clock
- **11**: System clock (SYSCLK) selected as ADC clock

#### Bits 27:26 CLK48SEL[1:0]: 48 MHz clock source selection
These bits are set and cleared by software to select the 48 MHz clock source used by USB and true RNG. The true RNG clock source selection is furthermore selected by RNGSEL.
- **00**: HSI48 clock selected as 48 MHz clock
- **01**: PLLSAI1 "Q" clock (PLLSAI1QCLK) selected as 48 MHz clock
- **10**: PLL "Q" clock (PLLQCLK) selected as 48 MHz clock
- **11**: MSI clock selected as 48 MHz clock

#### Bits 25:24 Reserved, must be kept at reset value.

#### Bits 23:22 SAI1SEL[1:0]: SAI1 clock source selection
These bits are set and cleared by software to select the SAI1 clock source.
- **00**: PLLSAI1 "P" clock (PLLSAI1PCLK) selected as SAI1 clock
- **01**: PLL "P" clock (PLLCLK) selected as SAI1 clock
- **10**: HSI16 clock selected as SAI1 clock
- **11**: External input SAI1_EXTCLK selected as SAI1 clock

**Caution:** If the selected clock is the external clock, it is not possible to switch to another clock if the external clock is not present.
8.4.30 RCC backup domain control register (RCC_BDCR)

Address offset: 0x090

Reset value: 0x0000 0000, (reset by Backup domain reset, except LSCOSEL, LSCOEN and BDRST, which are reset only by Backup domain power-on reset, not reset by wakeup from Standby and System 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 RCC backup 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 PWR control register 1 (PWR_CR1) has to be set before these can be modified. Refer to Section 6.1.4: Battery backup domain for further information. These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a Backup domain reset. Any internal or external Reset will not have any effect on these bits.

| Bit 31:26 | Reserved, must be kept at reset value. |
| Bit 25  | LSCOSEL: Low speed clock output selection |
| 0: LSI clock selected | 1: LSE clock selected |
| Bit 24  | LSCOEN: Low speed clock output enable |
| 0: Low speed clock output (LSCO) disable | 1: Low speed clock output (LSCO) enable |
| Bit 23:17 | Reserved, must be kept at reset value. |
| Bit 16  | BDRST: Backup domain software reset |
| 0: Reset not activated | 1: Resets the entire Backup domain |
| Bit 15  | RTCEN: RTC clock enable |
| 0: RTC clock disabled | 1: RTC clock enabled |
| Bit 14:10 | Reserved, must be kept at reset value. |
| Bit 9:8  | RTCSEL[1:0]: RTC clock source selection |
| 00: No clock | 01: LSE oscillator clock used as RTC clock |
| 10: LSI oscillator clock used as RTC clock | 11: HSE oscillator clock divided by 32 used as RTC clock |
| Bit 7  | Reserved, must be kept at reset value. |
8.4.31 RCC control/status register (RCC_CSR)

Address: 0x094

Reset value: 0x0C00 0000 (reset by System reset, except reset flags by POR only, not reset by wakeup from Standby)

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 has been detected by the clock security system on the external 32 kHz oscillator (LSE).
- 0: No failure detected on LSE (32 kHz oscillator)
- 1: Failure detected on LSE (32 kHz oscillator)

Bit 5 **LSECSSON** CSS on LSE enable
- Set by software to enable the clock security system on LSE (32 kHz oscillator).
- 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.
- 0: CSS on LSE (32 kHz external oscillator) OFF
- 1: CSS on LSE (32 kHz external oscillator) ON

Bits 4:3 **LSEDRV[1:0]** LSE oscillator drive capability
- Set by software to modulate the LSE oscillator drive capability.
- 00: ‘Xtal mode’ lower driving capability
- 01: ‘Xtal mode’ medium low driving capability
- 10: ‘Xtal mode’ medium high driving capability
- 11: ‘Xtal mode’ higher driving capability
- The oscillator is in Xtal mode when it is not in bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass
- Set and cleared by software to bypass oscillator. This bit can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).
- 0: LSE oscillator not bypassed
- 1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready
- Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after six external low-speed oscillator clock cycles.
- 0: LSE oscillator not ready
- 1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enable
- Set and cleared by software.
- 0: LSE oscillator OFF
- 1: LSE oscillator ON
Bit 31 **LPWRRSTF**: Low-power reset flag  
Set by hardware when a reset occurs due to illegal Stop, Standby or Shutdown mode entry.  
Cleared by writing to the RMVF bit.  
0: No illegal mode reset occurred  
1: Illegal mode reset occurred  

Bit 30 **WWDGRSTF**: Window watchdog reset flag  
Set by hardware when a window watchdog reset occurs.  
Cleared by writing to 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 writing to 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 writing to the RMVF bit.  
0: No software reset occurred  
1: Software reset occurred  

Bit 27 **BORRSTF**: BOR flag  
Set by hardware when a BOR occurs.  
Cleared by writing to the RMVF bit.  
0: No BOR occurred  
1: BOR occurred  

Bit 26 **PINRSTF**: Pin reset flag  
Set by hardware when a reset from the NRST pin occurs.  
Cleared by writing to 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 writing to 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.
Bit 23 **RMVF**: Remove reset flag
   Set by software to clear the reset flags.
   0: No effect
   1: Clear the reset flags

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **RFRSTS**: Radio system BLE and 802.15.4 reset status
   Set and cleared by hardware
   0: Radio system BLE and 802.15.4 not in reset, radio system can be accessed
   1: Radio system BLE and 802.15.4 under reset, radio system cannot be accessed

Bits 15:14 **RFWKPSEL[1:0]**: RF system wakeup clock source selection
   Set by software to select the clock source for RF system wakeup logic.
   00: No clock
   01: LSE oscillator clock used as RF system wakeup clock
   10: Reserved
   11: HSE oscillator clock divided by 1024 used as RF system wakeup clock

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:8 **LSI2TRIM[3:0]**: LSI2 oscillator trim.
   Note: LSI2TRIM must be changed only when LSI2 is disabled (LSI2ON = 0).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LSI2RDY**: LSI2 oscillator ready
   Set and cleared by hardware to indicate when the LSI2 oscillator is stable. After the LSI2ON bit is cleared, LSI2RDY goes low after three LSI2 oscillator clock cycles.
   0: LSI2 oscillator not ready
   1: LSI2 oscillator ready

Bit 2 **LSI2ON**: LSI2 oscillator enable and selection
   Set and cleared by software.
   0: LSI2 oscillator OFF (LSI1 selected on LSI)
   1: LSI2 oscillator ON (LSI2 when ready selected on LSI)

Bit 1 **LSI1RDY**: LSI1 oscillator ready
   Set and cleared by hardware to indicate when the LSI1 oscillator is stable. After the LSI1ON bit is cleared, LSI1RDY goes low after three LSI1 oscillator clock cycles. This bit can be set even if LSI1ON = 0 if the LSI1 is requested by the clock security system on LSE, by the Independent watchdog or by the RTC.
   0: LSI1 oscillator not ready
   1: LSI1 oscillator ready

Bit 0 **LSI1ON**: LSI1 oscillator enable
   Set and cleared by software.
   0: LSI1 oscillator OFF
   1: LSI1 oscillator ON

### 8.4.32 RCC clock recovery RC register (RCC_CRRCCR)

Address: 0x098

Reset value: 0x0000 XXX0 where X is factory-programmed.

Access: No wait state, word, half-word and byte access
## 8.4.33 RCC clock HSE register (RCC_HSECR)

**Address:** 0x09C  
**Reset value:** 0x0000 0030 (reset by System reset, not reset by wakeup from Standby)  
**Access:** Write access is protected and software must write a KEY (0xCAFE CAFE) as a word access in this register to unlock the register. Once unlocked a single write access (word, half-word or byte) can be performed to the register. It is then locked again. No wait states.

### Register Map

<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>

### Register Description

- **Bits 31:16** Reserved, must be kept at reset value
- **Bits 15:7** **HSETUNE[8:0]:** HSE clock calibration  
  These bits are initialized at startup with the factory-programmed HSI48 calibration trim value. They are ready only.
- **Bits 6:2** Reserved, must be kept at reset value
- **Bit 1** **HSETUNE[8:0]:** HSE clock enable  
  Set by hardware to indicate that HSI48 oscillator is stable. This bit is set only when HSI48 is enabled by software by setting HSI48ON.  
  - 0: HSI48 oscillator not ready  
  - 1: HSI48 oscillator ready
- **Bit 0** **HSETUNE[5:0]:** HSE capacitor tuning  
  Can be changed by software. Do not change when HSI48 is on.  
  - 0x00: minimum load capacitance  
  - 0x3F maximum load capacitance
- **Bits 7** Reserved, must be kept at reset value
8.4.34 RCC extended clock recovery register (RCC_EXTCFGR)

Address: 0x108
Reset value: 0x0003 0000.

Access: No wait state, word, half-word and byte access

Bits 6:4 HSEGMC[2:0]: HSE current control
Can be changed by software. Do not change when HSE is on.
000: current max limit 0.18 mA/V
001: current max limit 0.57 mA/V
010: current max limit 0.78 mA/V
011: current max limit 1.13 mA/V
100: current max limit 0.61 mA/V
101: current max limit 1.65 mA/V
110: current max limit 2.12 mA/V
111: current max limit 2.84 mA/V

Note: The HSEGMC value must be greater than g_mcorp whose value must be calculated according to AN2867 "Oscillator design guide for STM8S, STM8A and STM32 microcontrollers", available on www.st.com.

Bit 3 HSES: HSE Sense amplifier threshold
Can be changed by software. Do not change when HSE is on.
0: HSE bias current factor 1/2
1: HSE bias current factor 3/4

Bits 2:1 Reserved, must be kept at reset value

Bit 0 UNLOCKED: HSE clock control register unlocked
Set and cleared by hardware.
0: register locked, the key has not been programmed, or a write access has been performed to the register
1: Register unlocked. Key 0xCAFE CAFE has been written to unlock this register, enabling a single data write

Bits 31:21 Reserved, must be kept at reset value

Bit 20 RFCSS: Radio system HCLK5 and APB3 selected clock source indication
Set and reset by hardware to indicate which clock source is selected for the Radio system HCLK5 and APB3 clock.
0: HSI16 used for Radio system HCLK5 and APB3 clock
1: HSE divided by 2 used for Radio system HCLK5 and APB3 clock

Bits 19:18 Reserved, must be kept at reset value
Bit 17 **C2HPREF**: HCLK2 prescaler flag (CPU2)
- Set and reset by hardware to acknowledge HCLK2 prescaler programming
- Reset when a new prescaler value is programmed in C2HPRE. set when the programmed value is actually applied.
  - 0: HCLK2 prescaler value not yet applied
  - 1: HCLK2 prescaler value applied

Bit 16 **SHDHPREF**: HCLK4 shared prescaler flag (AHB4, Flash memory and SRAM2)
- Set and reset by hardware to acknowledge Shared HCLK4 prescaler programming
- Reset when a new prescaler value is programmed in SHDHPRE. set when the programmed value is actually applied.
  - 0: HCLK4 prescaler value not yet applied
  - 1: HCLK4 prescaler value applied

Bits 15:8 Reserved, must be kept at reset value

Bits 7:4 **C2PRE[3:0]**: HCLK2 prescaler (CPU2)
- Set and cleared by software to control the division factor of the HCLK2 clock (CPU2).
- The C2HPREF flag can be checked to know if the programmed C2HPRE prescaler value is applied.

**Caution:** Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to Section 6.1.6: Dynamic voltage scaling management). After a write operation to these bits and before decreasing the voltage range the register bit C2HPREF must be read to be sure that the new value has been taken into account.

- 0001: SYSCLK divided by 3
- 0010: SYSCLK divided by 5
- 0101: SYSCLK divided by 6
- 0110: SYSCLK divided by 10
- 0111: SYSCLK divided by 32
- 1000: SYSCLK divided by 2
- 1001: SYSCLK divided by 4
- 1010: SYSCLK divided by 8
- 1011: SYSCLK divided by 16
- 1100: SYSCLK divided by 64
- 1101: SYSCLK divided by 128
- 1110: SYSCLK divided by 256
- 1111: SYSCLK divided by 512
- Others: SYSCLK not divided
Bit 3:0 **SHDHPRE[3:0]**: HCLK4 shared prescaler (AHB4, Flash memory and SRAM2)
Set and cleared by software to control the division factor of the Shared HCLK4 clock (AHB4, Flash memory and SRAM2).
The SHDHPREF flag can be checked to know if the programmed SHDHPRE prescaler value is applied.

**Caution:** Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details refer to *Section 6.1.6: Dynamic voltage scaling management*). After a write operation to these bits and before decreasing the voltage range the register bit SHDHPRE must be read to be sure that the new value has been taken into account.

0001: SYSCLK divided by 3
0010: SYSCLK divided by 5
0101: SYSCLK divided by 6
0110: SYSCLK divided by 10
0111: SYSCLK divided by 32
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512
Others: SYSCLK not divided

### 8.4.35 RCC CPU2 AHB1 peripheral clock enable register (RCC_C2AHB1ENR)

Address offset: 0x148
Reset value: 0x0000 0000
Access: No wait state, word, half-word and byte access

**Note:** When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not 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>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 31:17 Reserved, must be kept at reset value.

Bit 16 **TSCEN**: CPU2 Touch sensing controller clock enable (STM32WB55xx only)
Set and cleared by software.
0: TSC clock disable for CPU2
1: TSC clock enable for CPU2
Note that this bit is reserved on STM32WB35xx.
8.4.36 RCC CPU2 AHB2 peripheral clock enable register (RCC_C2AHB2ENR)

Address offset: 0x14C
Reset value: 0x0000 0000
Access: No wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not 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></td>
<td></td>
<td></td>
<td></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>
8.4.37 RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC_C2AHB3ENR)

Address offset: 0x150
Reset value: 0x0208 0000
Access: No wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.
### Reset and clock control (RCC)

#### 8.4.38 RCC CPU2 APB1 peripheral clock enable register 1 (RCC_C2APB1ENR1)

<table>
<thead>
<tr>
<th>Bit 31-26</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 25    | **FLASHEN** CPU2 Flash interface clock enable  
This bit can only be cleared when the Flash is in Power down. Set and cleared by software.  
0: Flash interface clock disable for CPU2  
1: Flash interface clock enable for CPU2 |
| Bit 24:21 | Reserved, must be kept at reset value. |
| Bit 20    | **IPCCEN** CPU2 IPCC interface clock enable  
Set and cleared by software.  
0: IPCC clock disable for CPU2  
1: IPCC clock enable for CPU2 |
| Bit 19    | **HSEMEN** CPU2 HSEM clock enable  
Set and cleared by software.  
0: HSEM clock disable for CPU2  
1: HSEM clock enable for CPU2 |
| Bit 18    | **RNGEN** CPU2 true RNG clocks enable  
Set and cleared by software.  
0: True RNG bus and kernel clocks disable for CPU2  
1: True RNG bus and kernel clocks enable for CPU2 |
| Bit 17    | **AES2EN** CPU2 AES2 accelerator clock enable  
Set and cleared by software.  
0: AES2 clock disable for CPU2  
1: AES2 clock enable for CPU2 |
| Bit 16    | **PKAEN** CPU2 PKA accelerator clock enable  
Set and cleared by software.  
0: PKA clock disable for CPU2  
1: PKA clock enable for CPU2 |
| Bits 15:0 | Reserved, must be kept at reset value. |

**Address:** 0x158  
**Reset value:** 0x0000 0400  
**Access:** No wait state, word, half-word and byte access  
**Note:** When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not supported.
Bit 31 **LPTIM1EN**: CPU2 Low power timer 1 clocks enable
Set and cleared by software.
0: LPTIM1 bus and kernel clocks disabled for CPU2
1: LPTIM1 bus and kernel clocks enabled for CPU2

Bits 30:27 Reserved, must be kept at reset value.

Bit 26 **USBEN**: CPU2 USB FS clocks enable
Set and cleared by software.
0: USB FS bus and kernel clocks disabled for CPU2
1: USB FS bus and kernel clocks enabled for CPU2

Bits 25 Reserved, must be kept at reset value.

Bit 24 **CRSEN**: CPU2 CRS clock enable
Set and cleared by software.
0: CRS clock disabled for CPU2
1: CRS clock enabled for CPU2

Bit 23 **I2C3EN**: CPU2 I2C3 clocks enable
Set and cleared by software.
0: I2C3 bus and kernel clocks disabled for CPU2
1: I2C3 bus and kernel clocks enabled for CPU2

Bits 22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: CPU2 I2C1 clocks enable
Set and cleared by software.
0: I2C1 bus and kernel clocks disabled for CPU2
1: I2C1 bus and kernel clocks enabled for CPU2

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **SPI2EN**: CPU2 SPI2 clock enable (STM32WB55xx only)
Set and cleared by software.
0: SPI2 clock disabled for CPU2
1: SPI2 clock enabled for CPU2
Note that this bit is reserved on STM32WB35xx.

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

Bit 10 **RTCAPBEN**: CPU2 RTC APB clock enable
Set and cleared by software.
0: RTC APB clock disabled for CPU2
1: RTC APB clock enabled for CPU2
8.4.39 RCC CPU2 APB1 peripheral clock enable register 2 (RCC_C2APB1ENR2)

Address offset: 0x15C
Reset value: 0x00000
Access: No wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit 31:6 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 5  LPTIM2EN CPU2 Low power timer 2 clocks enable</td>
</tr>
<tr>
<td>Set and cleared by software.</td>
</tr>
<tr>
<td>0: LPTIM2 bus and kernel clocks disable for CPU2</td>
</tr>
<tr>
<td>1: LPTIM2 bus and kernel clocks enable for CPU2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM2EN</td>
</tr>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPUART1EN</td>
</tr>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

8.4.40 RCC CPU2 APB2 peripheral clock enable register (RCC_C2APB2ENR)

Address: 0x160
Reset value: 0x00000
Access: word, half-word and byte access
**Note:** When the peripheral clock is not active the peripheral registers read or write access from CPU2 is not supported.

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

**Bit 21** \( \text{SAI1EN}: \text{CPU2 SAI1 clocks enable} \)
Set and cleared by software.
0: SAI1 bus and kernel clocks disabled for CPU2
1: SAI1 bus and kernel clocks enabled for CPU2

**Bits 20:19** Reserved, must be kept at reset value.

**Bit 18** \( \text{TIM17EN}: \text{CPU2 TIM17 timer clock enable} \)
Set and cleared by software.
0: TIM17 timer clock disabled for CPU2
1: TIM17 timer clock enabled for CPU2

**Bit 17** \( \text{TIM16EN}: \text{CPU2 TIM16 timer clock enable} \)
Set and cleared by software.
0: TIM16 timer clock disabled for CPU2
1: TIM16 timer clock enabled for CPU2

**Bits 16:15** Reserved, must be kept at reset value.

**Bit 14** \( \text{USART1EN}: \text{CPU2 USART1 clock enable} \)
Set and cleared by software.
0: USART1 bus and kernel clocks disabled for CPU2
1: USART1 bus and kernel clocks enabled for CPU2

**Bit 13** Reserved, must be kept at reset value.

**Bit 12** \( \text{SPI1EN}: \text{CPU2 SPI1 clock enable} \)
Set and cleared by software.
0: SPI1 clock disabled for CPU2
1: SPI1 clock enabled for CPU2

**Bit 11** \( \text{TIM1EN}: \text{CPU2 TIM1 timer clock enable} \)
Set and cleared by software.
0: TIM1 timer clock disabled for CPU2
1: TIM1P timer clock enabled for CPU2

**Bits 10:0** Reserved, must be kept at reset value.

### 8.4.41 RCC CPU2 APB3 peripheral clock enable register (RCC_C2APB3ENR)

Address offset: 0x164

Reset value: 0x0000 0000
Access: No wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access from CPU2 is not 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>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, must be kept at reset value.

Bit 1 **802EN**: CPU2 802.15.4 interface clock enable
Set and cleared by software.
0: 802.15.4 clock disable for CPU2
1: 802.15.4 clock enable for CPU2

Bit 0 **BLEEN**: CPU2 BLE interface clock enable
Set and cleared by software.
0: BLE clock disable for CPU2
1: BLE clock enable for CPU2

**8.4.42 RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC_C2AHB1SMENR)**

Address offset: 0x168
Reset value: 0x0001 1007
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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>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 31:17 Reserved, must be kept at reset value.

Bit 16 **TSCSMEN**: Touch sensing controller clock enable during CPU2 CSleep mode (STM32WB55xx only)
Set and cleared by software.
0: TSC clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: TSC clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode
Note that this bit is reserved on STM32WB35xx.

Bits 15:13 Reserved, must be kept at reset value.
8.4.43 RCC CPU2 AHB2 peripheral clocks enable in Sleep modes register (RCC_C2AHB2SMENR)

Address offset: 0x16C
Reset value: 0x0001 209F
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>rw</td>
<td>rw</td>
<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 31:17 Reserved, must be kept at reset value.
Bit 16 **AES1SMEN**: AES1 accelerator clock enable during CPU2 CSleep mode
Set and cleared by software.
0: AES1 clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: AES1 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ADCSMEN**: ADC clock enable during CPU2 CSleep and CStop modes
Set and cleared by software.
0: ADC bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: ADC bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHSMEN**: IO port H clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port H clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: IO port H clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **GPIOESMEN**: IO port E clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port E clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: IO port E clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 3 **GPIODSMEN**: IO port D clock enable during CPU2 CSleep mode (STM32WB55xx only)
Set and cleared by software.
0: IO port D clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: IO port D clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.
Note that this bit is reserved on STM32WB35xx.

Bit 2 **GPIOCSMEN**: IO port C clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port C clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: IO port C clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 1 **GPIOBSMEN**: IO port B clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port B clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: IO port B clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 0 **GPIOASMEN**: IO port A clock enable during CPU2 CSleep mode
Set and cleared by software.
0: IO port A clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: IO port A clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.
8.4.44 RCC CPU2 AHB3 and AHB4 peripheral clocks enable in Sleep mode register (RCC_C2AHB3SMENR)

Address offset: 0x170

Reset value: 0x0307 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 31:26 Reserved, must be kept at reset value.

Bit 25 **FLASHSMEN** Flash memory interface clock enable during CPU2 CSleep mode
Set and cleared by software.
0: Flash memory interface clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: Flash memory interface clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 24 **SRAM2SMEN**: SRAM2a and SRAM2b interface clock enable during CPU2 CSleep mode
Set and cleared by software.
0: SRAM2 clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: SRAM2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **RNGSMEN** True RNG clock enable during CPU2 CSleep and CStop mode
Set and cleared by software.
0: True RNG bus clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: True RNG bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 17 **AES2SMEN** AES2 accelerator clock enable during CPU2 CSleep mode
Set and cleared by software.
0: AES2 clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: AES2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bit 16 **PKASMEN**: PKA accelerator clock enable during CPU2 CSleep mode
Set and cleared by software.
0: PKA clock disabled by the clock gating during CPU2 CSleep and CStop modes
1: PKA clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Bits 15:0 Reserved, must be kept at reset value.
8.4.45 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 1 (RCC_C2APB1SMENR1)

Address: 0x178
Reset value: 0x85A0 4601
Access: No wait state, word, half-word and byte access

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Access</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LPTIM1SMEN: Low power timer 1 clock enable during CPU2 CSleep and CStop mode</td>
<td></td>
<td>rw</td>
<td>0: LPTIM1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes. 1: LPTIM1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.</td>
</tr>
<tr>
<td>30:27</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>USBSMEN: USB FS clock enable during CPU2 CSleep and CStop modes</td>
<td></td>
<td>rw</td>
<td>0: USB FS bus clock disabled by the clock gating during CPU2 CSleep and CStop modes. 1: USB FS bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>CRSSMEN: CRS clock enable during CPU2 CSleep mode</td>
<td></td>
<td>rw</td>
<td>0: CRS clock disabled by the clock gating during CPU2 CSleep and CStop modes. 1: CRS clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.</td>
</tr>
<tr>
<td>23</td>
<td>I2C3SMEN: I2C3 clock enable during CPU2 CSleep and CStop modes</td>
<td></td>
<td>rw</td>
<td>0: I2C3 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes. 1: I2C3 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>I2C1SMEN: I2C1 clock enable during CPU2 CSleep and CStop modes</td>
<td></td>
<td>rw</td>
<td>0: I2C1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes. 1: I2C1 bus clock enabled by the clock gating during CPU2 CSleep modes disabled during CPU2 CStop mode.</td>
</tr>
<tr>
<td>20:15</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
8.4.46 RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 2 (RCC_C2APB1SMENR2)

Address offset: 0x17C
Reset value: 0x0000 0021
Access: No wait state, word, half-word and byte access

- Bit 14 SPI2SMEN: SPI2 clock enable during CPU2 CSleep mode (STM32WB55xx only)
  - Set and cleared by software.
  - 0: SPI2 clock disabled by the clock gating during CPU2 CSleep and CStop modes.
  - 1: SPI2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.
  - Note that this bit is reserved on STM32WB35xx.

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

- Bit 10 RTCAPBSMEN: RTC bus clock enable during CPU2 CSleep mode
  - Set and cleared by software.
  - 0: RTC bus clock disabled during by the clock gating during CPU2 CSleep and CStop modes.
  - 1: RTC bus clock enabled during by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

- Bit 9 LCDSMEN: LCD clock enable during CPU2 CSleep mode (STM32WB55xx only)
  - Set and cleared by software.
  - 0: LCD clock disabled by the clock gating during CPU2 CSleep and CStop modes.
  - 1: LCD clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.
  - Note that this bit is reserved on STM32WB35xx.

- Bits 8:1 Reserved, must be kept at reset value.

- Bit 0 TIM2SMEN: TIM2 timer clock enable during CPU2 CSleep mode
  - Set and cleared by software.
  - 0: TIM2 clock disabled by the clock gating during CPU2 CSleep and CStop modes.
  - 1: TIM2 clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

- Bit 5 LPTIM2SMEN Low power timer 2 clock enable during CPU2 CSleep and CStop modes.
  - Set and cleared by software.
  - 0: LPTIM2 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.
  - 1: LPTIM2 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.
8.4.47 RCC CPU2 APB2 peripheral clocks enable in Sleep mode register (RCC_C2APB2SMENR)

Address: 0x180
Reset value: 0x0026 5800
Access: 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>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>
<tr>
<td>Bit 15</td>
<td>Bit 14</td>
<td>Bit 13</td>
<td>Bit 12</td>
<td>Bit 11</td>
<td>Bit 10</td>
<td>Bit 9</td>
<td>Bit 8</td>
<td>Bit 7</td>
<td>Bit 6</td>
<td>Bit 5</td>
<td>Bit 4</td>
<td>Bit 3</td>
<td>Bit 2</td>
<td>Bit 1</td>
<td>Bit 0</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 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SMEN**: SAI1 clock enable during CPU2 CSleep and CStop mode
Set and cleared by software.
0: SAI1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: SAI1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM17 timer clock enable during CPU2 CSleep mode
Set and cleared by software.
0: TIM17 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: TIM17 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 17 **TIM16SMEN**: TIM16 timer clock enable during CPU2 CSleep mode
Set and cleared by software.
0: TIM16 timer clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: TIM16 timer clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN**: USART1 clock enable during CPU2 CSleep and CStop mode
Set and cleared by software.
0: USART1 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes.
1: USART1 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode.

Bit 13 Reserved, must be kept at reset value.
8.4.48 RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR)

Address offset: 0x184

Reset value: 0x00000 0003

Access: 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></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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><strong>802SMEN</strong></td>
</tr>
</tbody>
</table>

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

- **Bit 1 802SMEN:** 802.15.4 interface bus clock enable during CPU2 CSleep mode
  - Set and cleared by software.
  - 0: 802.15.4 bus clock disabled by the clock gating during CPU2 CSleep and CStop modes
  - 1: 802.15.4 bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

- **Bit 0 BLESSEN:** BLE interface bus clock enable during CPU2 CSleep mode
  - Set and cleared by software.
  - 0: BLE bus clock disabled by the clock gating during CPU2 CSleep and CStop modes
  - 1: BLE bus clock enabled by the clock gating during CPU2 CSleep mode, disabled during CPU2 CStop mode

Note: The BLE interface bus clock is also enabled by the BLE IP itself.
# RCC register map

The following table gives the RCC register map and the reset values.

## Table 42. RCC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>RCC_CCR</th>
<th>RCC_ICSCR</th>
<th>RCC_CFG</th>
<th>RCC_PLLCFG</th>
<th>RCC_PLLSAI1CFG</th>
<th>RCC_CICR</th>
<th>RCC_CIFR</th>
<th>RCC_CICR</th>
<th>RCC_SMPSCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</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 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>0x004</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>1 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>0x008</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 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>0x00C</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 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>0x010</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 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>0x014</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</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 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>0x01C</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 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>0x020</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 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>0x024</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>1 1 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>Offset</td>
<td>Register</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>
</tr>
<tr>
<td>0000h</td>
<td>RCC_AHB1RSTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0004h</td>
<td>RCC_AHB2RSTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008h</td>
<td>RCC_APB1RSTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000Ch</td>
<td>RCC_APB2RSTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010h</td>
<td>RCC_AHB1ENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0014h</td>
<td>RCC_AHB2ENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0018h</td>
<td>RCC_APB1ENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001Ch</td>
<td>RCC_APB2ENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020h</td>
<td>APB1RSTR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0024h</td>
<td>APB2RSTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028h</td>
<td>LPTIM1RST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>002Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0030h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0034h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0038h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0040h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0044h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0048h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>004Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0050h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0054h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0058h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>005Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0060h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0064h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0068h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0070h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0074h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0078h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0080h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0084h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0088h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0090h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0094h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0098h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009Ch</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A0h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A4h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A8h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00Ac</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B0h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B4h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B8h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00BCh</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C0h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C4h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C8h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00Cc</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D0h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D4h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D8h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00DCh</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00E0h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00E4h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00E8h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00Ec</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00F0h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00F4h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00F8h</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00Fc</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 42: RCC register map and reset values (continued)
Table 42. RCC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>0x054</th>
<th>0x058</th>
<th>0x05C</th>
<th>0x060</th>
<th>0x064</th>
<th>0x068</th>
<th>0x06C</th>
<th>0x070</th>
<th>0x074</th>
<th>0x078</th>
<th>0x07C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RCC_AHB1SMENR</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>LPTIM2EN</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>USBEN</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>SEREN</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>USBEN</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>RCC_APB1ENR1</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>RCC_APB1ENR2</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>RCC_APB2ENR</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>_APB1SMENR1</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>_APB1SMENR2</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>_AHB3SMENR</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>AHB2SMENR</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>AHB1SMENR</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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>RCC_APB1ENR1</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>RCC_APB1ENR2</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>RCC_APB2ENR</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>_APB1SMENR1</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>_APB1SMENR2</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>_AHB3SMENR</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>AHB2SMENR</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>AHB1SMENR</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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>RCC_AHB1SMENR</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>LPTIM2EN</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>USBEN</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>SEREN</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>USBEN</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>RCC_APB1ENR1</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>RCC_APB1ENR2</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>RCC_APB2ENR</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>_APB1SMENR1</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>_APB1SMENR2</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>_AHB3SMENR</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>AHB2SMENR</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>AHB1SMENR</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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>RCC_APB1ENR1</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>RCC_APB1ENR2</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>RCC_APB2ENR</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>_APB1SMENR1</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>_APB1SMENR2</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>_AHB3SMENR</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>AHB2SMENR</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>AHB1SMENR</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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 42. 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  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x080  | RCC_APB2SMENR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x084  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x088  | RCC_CCIPR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x08C  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x090  | RCC_BDCR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x094  | RCC_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  | 0  | 0  | 0  |
| 0x098  | RCC_CRRCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x09C  | RCC_HSECR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0A0  | to 0x104       | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x108  | RCC_EXTCFGR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10C  | to 0x144       | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x148  | RCC_C2AHB1ENR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table continues...
| Offset | Register | 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 |
|--------|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x14C  | RCC _C2AHB2ENR |    |    |    |    | 0  |    |    | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x150  | RCC _C2AHB3ENR |    |    |    |    | 0  |    |    | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x154  | Reserved      |    |    |    |    | 0  |    |    | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x158  | RCC _C2APB1ENR1 |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x15C  | RCC _C2APB1ENR2 |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x160  | RCC _C2APB2ENR |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x164  | RCC _C2APB3ENR |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 0  |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x168  | RCC _C2AHB1SMENR |    |    |    |    | 1  |    |    | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 1  |    |    | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x16C  | RCC _C2AHB2SMENR |    |    |    |    | 1  |    |    | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 1  |    |    | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x170  | RCC _C2AHB3SMENR |    |    |    |    | 1  |    |    | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Offset Register |    |    |    |    | 1  |    |    | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x174  | Reserved      |    |    |    |    | 1  |    |    | 0  | 0  | 1  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 42. RCC register map and reset values (continued)
Refer to Section 2.2 on page 66 for the register boundary addresses.
9 General-purpose I/Os (GPIO)

9.1 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).

9.2 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
- 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

9.3 GPIO implementation

<table>
<thead>
<tr>
<th>Table 43. GPIO implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>Port A</td>
</tr>
<tr>
<td>Port B</td>
</tr>
<tr>
<td>Port C</td>
</tr>
<tr>
<td>Port D</td>
</tr>
<tr>
<td>Port E</td>
</tr>
<tr>
<td>Port H</td>
</tr>
</tbody>
</table>
9.4 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 register 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 20 shows the basic structure of a Three-volt tolerant (TT) and Five-volt tolerant (FT) I/O port bit.

![Figure 20. Three-volt or five-volt tolerant GPIO structure (TT or FT)](image)

Note: The parasitic diode in the analog domain is connected to $V_{DDA}$ and cannot be used as a protection diode.

The voltage level called $V_{DD, FT}$ in some datasheets and reference manuals is inside the ESD protection block.

When the analog option is selected, the FT I/O is not five-volt tolerant anymore since the pin is supplied with $V_{DDA}$. 
A TT or FT GPIO pin has no internal protection diode connected to supply (V_DD). There is no physical limitation against over-voltage. Therefore, for applications requiring a limited voltage threshold, it is recommended to connect an external diode to V_DD.

*Table 44* gives the possible port bit configurations.

<table>
<thead>
<tr>
<th>MODE(i) [1:0]</th>
<th>OTYPER(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>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>0</td>
<td>0</td>
<td>0</td>
<td>AF output PP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>AF output PP + PU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>AF 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>AF output OD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>AF output OD + PU</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>AF output OD + PD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>00</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.
9.4.1 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:
- PA15: JTDI in input mode with pull-up
- PA14: JTCK/SWCLK in input mode with pull-down
- PA13: JTMS/SWDAT in input mode with pull-up
- PB4: NJTRST in input mode with pull-up
- PB3: JTDO in Hi-Z mode no pulls

PH3/BOOT0 is in input mode during the reset until at least the end of the option byte loading phase. See Section 9.4.15: Using PH3 as GPIO.

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.

9.4.2 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.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) 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:
– For the ADC, PVD and COMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and COMP registers.
– For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, 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.

9.4.3 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.

9.4.4 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.

See Section 9.5.5 and Section 9.5.6 for the register descriptions.

9.4.5 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.
9.4.6 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 9.5.8) 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 9.5.8.

9.4.7 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.

9.4.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.

Refer to Section 14: Extended interrupt and event controller (EXTI) and to Section 14.4: EXTI functional description.

9.4.9 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 21 shows the input configuration of the I/O port bit.
9.4.10 **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 22* shows the output configuration of the I/O port bit.
9.4.11 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

Note: The alternate function configuration described above is not applied when the selected alternate function is an LCD function. In this case, the I/O, programmed as an alternate function output, is configured as described in the analog configuration. Figure 23 shows the Alternate function configuration of the I/O port bit.

Figure 23. Alternate function configuration

9.4.12 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 24 shows the high-impedance, analog-input configuration of the I/O port bits.
9.4.13 Using the LSE oscillator pins as GPIOs

When the LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the LSE oscillator is switched ON (by setting the 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 OSC32_IN pin is reserved for clock input and the OSC32_OUT pin can still be used as normal GPIO.

Note: The HSE OSC_IN and OSC_OUT pins are dedicated oscillator pins and cannot be used as GPIO.

9.4.14 Using the GPIO pins in the RTC supply 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 Section 29.4: RTC functional description.

9.4.15 Using PH3 as GPIO

PH3 may be used as boot pin (BOOT0) or as a GPIO. Depending on the nSWBOOT0 bit in the user option byte, it switches from the input mode to the analog input mode:

- After the option byte loading phase if nSWBOOT0 = 1.
- After reset if nSWBOOT0 = 0.
9.5 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 45.

The peripheral registers can be written in word, half word or byte mode.

9.5.1 GPIO port mode register (GPIOx_MODER) (x = A to E and H)

Address offset: 0x00

Reset value: 0xABFF FFFF (for port A)
Reset value: 0xFFFF FEBF (for port B)
Reset value: 0xFFFF FFFF for port C, D
Reset value: 0x0000 03FF for port E
Reset value: 0x0000 00CF for port H

Port E[31:10] are reserved
Port H[31:8, 5:4] are reserved

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>MODE[15:0][1:0]: Port x configuration I/O pin y (y = 15 to 0)</td>
</tr>
<tr>
<td></td>
<td>These bits are written by software to configure the I/O mode.</td>
</tr>
<tr>
<td></td>
<td>00: Input mode</td>
</tr>
<tr>
<td></td>
<td>01: General purpose output mode</td>
</tr>
<tr>
<td></td>
<td>10: Alternate function mode</td>
</tr>
<tr>
<td></td>
<td>11: Analog mode (reset state)</td>
</tr>
</tbody>
</table>
9.5.2  GPIO port output type register (GPIOx_OTYPER)  
(x = A to E and H)  

Address offset: 0x04  
Reset value: 0x0000 0000  
Port E[31:5] are reserved  
Port H[31:4, 2] are reserved

<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>Ot5</td>
<td>Ot4</td>
<td>Ot13</td>
<td>Ot12</td>
<td>Ot11</td>
<td>Ot10</td>
<td>Ot9</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>
<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 31:16  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

9.5.3  GPIO port output speed register (GPIOx_OSPEEDR)  
(x = A to E and H)  

Address offset: 0x08  
Reset value: 0x0C00 0000 (for port A)  
Reset value: 0x0000 00C0 (for port B)  
Reset value: 0x0000 0000 (for other ports)  
Port E[31:10] are reserved  
Port H[31:8, 5:4] are reserved

<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>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></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>OSPEED7</td>
<td>OSPEED6</td>
<td>OSPEED5</td>
<td>OSPEED4</td>
<td>OSPEED3</td>
<td>OSPEED2</td>
<td>OSPEED1</td>
<td>OSPEED0</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
9.5.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR)
(x = A to E and H)

Address offset: 0x0C
Reset value: 0x6400 0000 (for port A)
Reset value: 0x0000 0100 (for port B)
Reset value: 0x0000 0000 (for other ports)
Port E[31:10] are reserved
Port H[31:8, 5:4] are reserved

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUPD15[1:0]</td>
<td>Port configuration I/O pin y (y = 15 to 0)</td>
</tr>
<tr>
<td>These bits are written by software to configure the I/O pull-up or pull-down</td>
<td></td>
</tr>
<tr>
<td>00: No pull-up, pull-down</td>
<td></td>
</tr>
<tr>
<td>01: Pull-up</td>
<td></td>
</tr>
<tr>
<td>10: Pull-down</td>
<td></td>
</tr>
<tr>
<td>11: Reserved</td>
<td></td>
</tr>
</tbody>
</table>

9.5.5 GPIO port input data register (GPIOx_IDR)
(x = A to E and H)

Address offset: 0x10
Reset value: 0x0000 XXXX
Port E[31:5] are reserved
Port H[31:4, 2] are reserved
9.5.7 GPIO port bit set/reset register (GPIOx_BSRR)  
(x = A to E and H)  
Address offset: 0x18  
Reset value: 0x0000 0000  
Port E[31:21, 15:5] are reserved  
Port H[31:20, 18, 15:4, 2] are reserved

Note: For atomic bit set/reset, OD bits can be individually set and/or reset by writing to GPIO port bit set/reset register (GPIOx_BSRR) (x = A to E and H).
9.5.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to E and H)

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
Port E[31:17, 15:5] are reserved
Port H[31:17, 15:4, 2] are reserved

<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></td>
<td></td>
<td></td>
<td></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>LCK15</td>
<td>LCK14</td>
<td>LCK13</td>
<td>LCK12</td>
<td>LCK11</td>
<td>LCK10</td>
<td>LCK9</td>
<td>LCK8</td>
<td>LCK7</td>
<td>LCK6</td>
<td>LCK5</td>
<td>LCK4</td>
<td>LCK3</td>
<td>LCK2</td>
<td>LCK1</td>
<td>LCK0</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>
9.5.9 GPIO alternate function low register (GPIOx_AFRL)  
(x = A to E and H)

Address offset: 0x20
Reset value: 0x0000 0000
Port E[31:20] are reserved
Port H[31:16, 11:8] are reserved

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

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

Bit 16 **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 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
Bits 31:0  **AFSEL[7:0][3:0]**: Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
- 0000: AF0
- 0001: AF1
- 0010: AF2
- 0011: AF3
- 0100: AF4
- 0101: AF5
- 0110: AF6
- 0111: AF7
- 1000: AF8
- 1001: AF9
- 1010: AF10
- 1011: AF11
- 1100: AF12
- 1101: AF13
- 1110: AF14
- 1111: AF15

9.5.10  **GPIO alternate function high register (GPIOx_AFRH)**
(x = A to E and H)

Address offset: 0x24
Reset value: 0x0000 0000
Port E[31:0] are reserved
Port H[31:0] are reserved

<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></td>
<td></td>
<td></td>
<td></td>
<td></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>

306/1530  RM0434 Rev 13
Bits 31:0  **AFSEL[15:8][3:0]**: Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
- 0000: AF0
- 0001: AF1
- 0010: AF2
- 0011: AF3
- 0100: AF4
- 0101: AF5
- 0110: AF6
- 0111: AF7
- 1000: AF8
- 1001: AF9
- 1010: AF10
- 1011: AF11
- 1100: AF12
- 1101: AF13
- 1110: AF14
- 1111: AF15

### 9.5.11  GPIO port bit reset register (GPIOx_BRR) (x = A to E and H)

**Address offset:** 0x28  
**Reset value:** 0x0000 0000  
Port E[31:5] are reserved  
Port H[31:4, 2] are reserved

<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>BR15</td>
<td>BR14</td>
<td>BR13</td>
<td>BR12</td>
<td>BR11</td>
<td>BR10</td>
<td>BR9</td>
<td>BR8</td>
<td>BR7</td>
<td>BR6</td>
<td>BR5</td>
<td>BR4</td>
<td>BR3</td>
<td>BR2</td>
<td>BR1</td>
<td>BR0</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>

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

Bits 15:0  **BR[15:0]**: Port x reset I/O 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
9.5.12 GPIO register map

The following table gives the GPIO 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   | GPIOA_MODER      | 1  | 0  | 0  | 0  | 1  | 0  | 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  |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00   | GPIOB_MODER      | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00   | GPIOx_MODER      | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
| (where x = C, D) |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00   | GPIOE_MODER      | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x00   | GPIOH_MODER      | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x04   | GPIOx_OTYPER     | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
| (where x = A..D) |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | GPIOE_OTYPER     | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x04   | GPIOH_OTYPER     | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x08   | GPIOA_OSPEEDR    | 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  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x08   | GPIOB_OSPEEDR    | 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  | 1  | 1  | 1  | 1  | 1  | 1  |

Reset value

308/1530 RM0434 Rev 13
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Table 45. GPIO register map and reset values (continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0C</td>
<td>GPIOA_PUPDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOA_IDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOA_ODR</td>
<td>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</td>
</tr>
<tr>
<td>0x10</td>
<td>GPIOB_PUPDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOB_IDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOB_ODR</td>
<td>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</td>
</tr>
<tr>
<td>0x18</td>
<td>GPIOC_PUPDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOC_IDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOC_ODR</td>
<td>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</td>
</tr>
<tr>
<td>0x1C</td>
<td>GPIOD_PUPDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOD_IDR</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>GPIOD_ODR</td>
<td>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</td>
</tr>
</tbody>
</table>

Note: The table continues with similar entries for other GPIO registers (GPIOH_PUPDR, GPIOx_ODR, GPIOx_IDR, Reset values) as per the given format.
### Table 45. GPIO 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   |
|--------|--------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x14   | GPIOE_ODR                      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | GPIOH_ODR                      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18   | GPIOx_BSRR ((where x = A..D))  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18   | GPIOE_BSRR                     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1C   | GPIOx_LCKR ((where x = A..D))  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1C   | GPIOH_LCKR                     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x20   | GPIOx_AFRL ((where x = A..D))  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x20   | GPIOH_AFRL                     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x24   | GPIOx_AFRH ((where x = A..D))  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x28   | GPIOx_BRR ((where x = A..D))   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x28   | GPIOH_BRR                      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Refer to Section 2.2 on page 66 for the register boundary addresses.

310/1530  RM0434 Rev 13
10  System configuration controller (SYSCFG)

10.1  SYSCFG main features

The devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Remapping memory areas
- Managing the external interrupt line connection to the GPIOs
- Managing robustness feature
- Setting SRAM2 and PKA RAM write protection and software erase
- Configuring FPU interrupts
- Enabling /disabling I2C Fast-mode Plus driving capability on some I/Os and voltage booster for I/Os analog switches
- Interrupt pre-masking

10.2  SYSCFG registers

10.2.1  SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap.

Address offset: 0x000

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT0 pin and BOOT1 option bit)

<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>
</table>

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

Bits 2:0  MEM_MODE[2:0]: Memory mapping selection

These bits control the memory internal mapping at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT mode setting. After reset these bits take the value selected by BOOT0 (pin or option bit depending on nSWBOOT0 option bit) and BOOT1 option bit.

000: Main Flash memory mapped at CPU1 0x00000000
001: System Flash memory mapped at CPU1 0x00000000
010: Reserved
011: SRAM1 mapped at CPU1 0x00000000
100: Reserved
101: Reserved
110: QUADSPI memory mapped at CPU1 0x00000000
111: Reserved
### 10.2.2 SYSCFG configuration register 1 (SYSCFG_CFGR1)

**Address offset:** 0x004  
**Reset value:** 0x7C00 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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>

**FPU_IE[5:0]:** CPU1 FPU interrupts enable bits  
- **FPU_IE[5]:** Inexact interrupt enable  
- **FPU_IE[4]:** Input denormal interrupt enable  
- **FPU_IE[3]:** Overflow interrupt enable  
- **FPU_IE[2]:** Underflow interrupt enable  
- **FPU_IE[1]:** Divide-by-zero interrupt enable  
- **FPU_IE[0]:** Invalid operation interrupt enable

**Bits 25:23** Reserved, must be kept at reset value.

**Bit 22** **I2C3_FMP:** I2C3 Fast-mode Plus driving capability activation  
This bit enables the Fm+ driving mode on I2C3 pins selected through AF selection bits.  
- 0: Fm+ mode is not enabled on I2C3 pins selected through AF selection bits  
- 1: Fm+ mode is enabled on I2C3 pins selected through AF selection bits

**Bit 21** Reserved, must be kept at reset value.

**Bit 20** **I2C1_FMP:** I2C1 Fast-mode Plus driving capability activation  
This bit enables the Fm+ driving mode on I2C1 pins selected through AF selection bits.  
- 0: Fm+ mode is not enabled on I2C1 pins selected through AF selection bits  
- 1: Fm+ mode is enabled on I2C1 pins selected through AF selection bits

**Bit 19** **I2C_PB9_FMP:** Fast-mode Plus (Fm+) driving capability activation on PB9  
This bit enables the Fm+ driving mode for PB9.  
- 0: PB9 pin operates in standard mode.  
- 1: Fm+ mode enabled on PB9 pin, and the Speed control is bypassed.

**Bit 18** **I2C_PB8_FMP:** Fast-mode Plus (Fm+) driving capability activation on PB8  
This bit enables the Fm+ driving mode for PB8.  
- 0: PB8 pin operates in standard mode.  
- 1: Fm+ mode enabled on PB8 pin, and the Speed control is bypassed.

**Bit 17** **I2C_PB7_FMP:** Fast-mode Plus (Fm+) driving capability activation on PB7  
This bit enables the Fm+ driving mode for PB7.  
- 0: PB7 pin operates in standard mode.  
- 1: Fm+ mode enabled on PB7 pin, and the Speed control is bypassed.

**Bit 16** **I2C_PB6_FMP:** Fast-mode Plus (Fm+) driving capability activation on PB6  
This bit enables the Fm+ driving mode for PB6.  
- 0: PB6 pin operates in standard mode.  
- 1: Fm+ mode enabled on PB6 pin, and the Speed control is bypassed.
Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable
- 0: I/O analog switches are supplied by V_{DDA} voltage. This is the recommended configuration when using the ADC in high V_{DDA} voltage operation.
- 1: I/O analog switches are supplied by a dedicated voltage booster (supplied by V_{DD}). This is the recommended configuration when using the ADC in low V_{DDA} voltage operation.

Bits 7:0 Reserved, must be kept at reset value.

---

**10.2.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)**

Address offset: 0x008

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>EXTI[3][2:0]</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

Bits 14:12 **EXTI3[2:0]**: EXTI 3 configuration bits
- These bits are written by software to select the source input for the EXTI3 external interrupt.
  - 000: PA[3] pin
  - 001: PB[3] pin
  - 010: PC[3] pin (STM32WB55xx only)
  - 100: PE[3] pin (STM32WB55xx only)
  - 101: Reserved
  - 110: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI2[2:0]**: EXTI 2 configuration bits
- These bits are written by software to select the source input for the EXTI2 external interrupt.
  - 000: PA[2] pin
  - 001: PB[2] pin
  - 010: PC[2] pin (STM32WB55xx only)
  - 100: PE[2] pin (STM32WB55xx only)
  - 101: Reserved
  - 110: Reserved
  - 111: Reserved

Bit 7 Reserved, must be kept at reset value.
10.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:15</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 14:12</td>
<td><strong>EXTI7[2:0]</strong>: EXTI 7 configuration bits</td>
</tr>
</tbody>
</table>

| Bits 6:4 | **EXTI1[2:0]**: EXTI 1 configuration bits |

These bits are written by software to select the source input for the EXTI interrupt.

- 000: PA[1] pin
- 001: PB[1] pin
- 010: PC[1] pin (STM32WB55xx only)
- 011: PD[1] pin (STM32WB55xx only)
- 100: PE[1] pin (STM32WB55xx only)
- 101: Reserved
- 110: Reserved
- 111: PH[1] pin (STM32WB55xx only)

Note: Some of the I/O pins mentioned in this register may be not available on small packages.
Bit 11  Reserved, must be kept at reset value.

Bits 10:8  **EXTI6[2:0]: EXTI 6 configuration bits**
These bits are written by software to select the source input for the EXTI6 external interrupt.
  000: PA[6] pin
  001: PB[6] pin
  010: PC[6] pin (STM32WB55xx only)
  100: Reserved
  101: Reserved
  110: Reserved
  111: Reserved

Bit 7  Reserved, must be kept at reset value.

Bits 6:4  **EXTI5[2:0]: EXTI 5 configuration bits**
These bits are written by software to select the source input for the EXTI5 external interrupt.
  000: PA[5] pin
  001: PB[5] pin
  010: PC[5] pin (STM32WB55xx only)
  100: Reserved
  101: Reserved
  110: Reserved
  111: Reserved

Bit 3  Reserved, must be kept at reset value.

Bits 2:0  **EXTI4[2:0]: EXTI 4 configuration bits**
These bits are written by software to select the source input for the EXTI4 external interrupt.
  000: PA[4] pin
  001: PB[4] pin
  010: PC[4] pin (STM32WB55xx only)
  100: PE[4] pin
  101: Reserved
  110: Reserved
  111: Reserved

*Note:* Some of the I/O pins mentioned in this register may be not available on small packages.

### 10.2.5  **SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)**

Address offset: 0x010

Reset value: 0x0000 0000
Bits 31:15  Reserved, must be kept at reset value.

Bits 14:12  **EXTII1[2:0]**: EXTI 11 configuration bits

These bits are written by software to select the source input for the EXTI11 external interrupt.
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 11  Reserved, must be kept at reset value.

Bits 10:8  **EXTI10[2:0]**: EXTI 10 configuration bits

These bits are written by software to select the source input for the EXTI10 external interrupt.
- 000: PA[10] pin
- 001: PB[10] pin (STM32WB55xx only)
- 010: PC[10] pin (STM32WB55xx only)
- 011: PD[10] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 7  Reserved, must be kept at reset value.

Bits 6:4  **EXTI9[2:0]**: EXTI 9 configuration bits

These bits are written by software to select the source input for the EXTI9 external interrupt.
- 000: PA[9] pin
- 001: PB[9] pin
- 010: PC[9] pin (STM32WB55xx only)
- 011: PD[9] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 3  Reserved, must be kept at reset value.
Bits 2:0 **EXTI8[2:0]**: EXTI 8 configuration bits

These bits are written by software to select the source input for the EXTI8 external interrupt.

- 000: PA[8] pin
- 001: PB[8] pin
- 010: PC[8] pin (STM32WB55xx only)
- 011: PD[8] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

**Note:** Some of the I/O pins mentioned in this register may be not available on small packages.

### 10.2.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

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>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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 14:12 **EXTI15[2:0]**: EXTI15 configuration bits

These bits are written by software to select the source input for the EXTI15 external interrupt.

- 000: PA[15] pin
- 001: PB[15] pin (STM32WB55xx only)
- 010: PC[15] pin
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **EXTI14[2:0]**: EXTI14 configuration bits

These bits are written by software to select the source input for the EXTI14 external interrupt.

- 000: PA[14] pin
- 001: PB[14] pin (STM32WB55xx only)
- 010: PC[14] pin
- 011: PD[14] pin (STM32WB55xx only)
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved
10.2.7 SYSCFG SRAM2 control and status register (SYSCFG_SCSR)

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>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>r</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **C2RFD**: CPU2 SRAM fetch (execution) disable.

This bit can be set by software and be reset by a hardware reset, including a reset after Standby. Software writing 'b0 has no effect.

0: CPU2 fetch from SRAM1, SRAM2a and SRAM2b enabled, allows CPU2 to fetch and execute code from SRAMs
1: CPU2 fetch from SRAM1, SRAM2a and SRAM2b disabled, Any CPU2 fetch from SRAMs generates a bus error.

Bits 30:2 Reserved, must be kept at reset value.
Bit 1 **SRAM2BSY**: SRAM2 and PKA RAM busy by erase operation
- 0: Nor SRAM2 neither PKA RAM erase operation is on going.
- 1: SRAM2 and/or PKA RAM erase operation is on going.

Bit 0 **SRAM2ER**: SRAM2 and PKA RAM Erase
Setting this bit starts a hardware SRAM2 and PKA RAM erase operation. This bit is automatically cleared at the end of the SRAM2 and PKA RAM erase operation.

*Note: This bit is write-protected: setting this bit is possible only after the correct key sequence is written in the SYSCFG_SKR register.*

### 10.2.8 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x01C
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>
<tr>
<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>
<td></td>
<td>rs</td>
<td>rs</td>
</tr>
</tbody>
</table>

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

**Bit 8 SPF**: SRAM2 parity error flag
This bit is set by hardware when an SRAM2 parity error is detected. It is cleared by software by writing '1'.
- 0: No SRAM2 parity error detected
- 1: SRAM2 parity error detected

Bits 7:4 Reserved, must be kept at reset value.

**Bit 3 ECCL**: ECC Lock
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Flash ECC error connection to TIM1/16/17 Break input.
- 0: ECC error disconnected from TIM1/16/17 Break input.
- 1: ECC error connected to TIM1/16/17 Break input.

**Bit 2 PVDL**: PVD lock enable bit
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR_CR2 register.
- 0: PVD interrupt disconnected from TIM1/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application.
- 1: PVD interrupt connected to TIM1/16/17 Break input, PVDE and PLS[2:0] bits are read only.

**Bit 1 SPL**: SRAM2 parity lock bit
This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/16/17 Break inputs.
- 0: SRAM2 parity error signal disconnected from TIM1/16/17 Break inputs.
- 1: SRAM2 parity error signal connected to TIM1/16/17 Break inputs
10.2.9  **SYSCFG SRAM2 write protection register (SYSCFG_SWPR1)**

Address offset: 0x020

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>P31WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>30</td>
<td>P30WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>29</td>
<td>P29WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>28</td>
<td>P28WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>27</td>
<td>P27WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>26</td>
<td>P26WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>25</td>
<td>P25WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>24</td>
<td>P24WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>23</td>
<td>P23WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>22</td>
<td>P22WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>21</td>
<td>P21WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>20</td>
<td>P20WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>19</td>
<td>P19WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>18</td>
<td>P18WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>17</td>
<td>P17WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
<tr>
<td>16</td>
<td>P16WP (PxWP, x = 0 to 31): SRAM2 1Kbyte page x write protection</td>
</tr>
</tbody>
</table>

Reserved

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

Bits 7:0  **KEY[7:0]: SRAM2 write protection key for software erase**

The following steps are required to unlock the write protection of the SRAM2ER bit in the SYSCFG_CFGR2 register.

1. Write 0xCA into Key[7:0]
2. Write 0x53 into Key[7:0]

Writing a wrong key reactivates the write protection.

10.2.10  **SYSCFG SRAM2 key register (SYSCFG_SKR)**

Address offset: 0x024

System 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>
</tbody>
</table>

KEY[7:0] w w w w w w w w

Bits 7:0  **KEY[7:0]: SRAM2 write protection key for software erase**

The following steps are required to unlock the write protection of the SRAM2ER bit in the SYSCFG_CFGR2 register.

1. Write 0xCA into Key[7:0]
2. Write 0x53 into Key[7:0]

Writing a wrong key reactivates the write protection.

10.2.11  **SYSCFG SRAM2 write protection register 2 (SYSCFG_SWPR2)**

Address offset: 0x028

System reset value: 0x0000 0000
Bits 31:0  **PxWP** (x= 32 to 63): SRAM2 1 Kbyte page x write protection

These bits are set by software and cleared only by a system reset.

0: Write protection of SRAM2 1 Kbyte page x is disabled.
1: Write protection of SRAM2 1 Kbyte page x is enabled.

### 10.2.12  SYSCFG CPU1 interrupt mask register 1 (SYSCFG_IMR1)

**Address offset:** 0x100

**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>EXT15 IM</td>
<td>EXT14 IM</td>
<td>EXT13 IM</td>
<td>EXT12 IM</td>
<td>EXT11IM</td>
<td>EXT10 IM</td>
<td>EXT9 IM</td>
<td>EXT8 IM</td>
<td>EXT7 IM</td>
<td>EXT6 IM</td>
<td>EXT5 IM</td>
<td>EXT4 IM</td>
<td>Ext3 IM</td>
<td>Ext2 IM</td>
<td>Ext1 IM</td>
<td>Ext0 IM</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>
<tr>
<td>TIM17 IM</td>
<td>TIM16 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</td>
<td>TIM1 IM</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 31:21  **xxxIM:** Peripheral xxx interrupt mask to CPU1

0: Peripheral xxx interrupt forwarded to CPU1
1. Peripheral xxx interrupt to CPU1 masked.

Bits 20:16  Reserved, must be kept at reset value.

Bits 15:13  **xxxIM:** Peripheral xxx interrupt mask to CPU1

0: Peripheral xxx interrupt forwarded to CPU1
1. Peripheral xxx interrupt to CPU1 masked.

Bits 12:0  Reserved, must be kept at reset value.

### 10.2.13  SYSCFG CPU1 interrupt mask register 2 (SYSCFG_IMR2)

**Address offset:** 0x104

**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>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>
<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>

**ST**
Bits 31:21: Reserved, must be kept at reset value.

Bit 20: **xxxIM**: Peripheral xxx interrupt mask to CPU1
0: Peripheral xxx interrupt forwarded to CPU1
1: Peripheral xxx interrupt to CPU1 masked.

Bit 19: Reserved, must be kept at reset value.

Bit 18: **xxxIM**: Peripheral xxx interrupt mask to CPU1
0: Peripheral xxx interrupt forwarded to CPU1
1: Peripheral xxx interrupt to CPU1 masked.

Bit 17: Reserved, must be kept at reset value.

Bits 16: **xxxIM**: Peripheral xxx interrupt mask to CPU1
0: Peripheral xxx interrupt forwarded to CPU1
1: Peripheral xxx interrupt to CPU1 masked.

Bits 15:0: Reserved, must be kept at reset value.

### 10.2.14 SYSCFG CPU2 interrupt mask register 1 (SYSCFG_C2IMR1)

Address offset: 0x108

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th><strong>xxxIM</strong>: Peripheral xxx interrupt mask to CPU2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Peripheral xxx interrupt forwarded to CPU2</td>
</tr>
<tr>
<td>1:</td>
<td>Peripheral xxx interrupt to CPU2 masked.</td>
</tr>
</tbody>
</table>

Bits 15:13: Reserved, must be kept at reset value.

Bits 12:8: **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1: Peripheral xxx interrupt to CPU2 masked.

Bit 7: Reserved, must be kept at reset value.

Bits 6:3: **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1: Peripheral xxx interrupt to CPU2 masked.

Bits 2:1: Reserved, must be kept at reset value.

Bit 0: **xxxIM**: Peripheral xxx interrupt mask to CPU2
0: Peripheral xxx interrupt forwarded to CPU2
1: Peripheral xxx interrupt to CPU2 masked.
10.2.15 SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2)

Address offset: 0x10C
System reset value: 0x0000 0000

| Bits 31:23 | Reserved, must be kept at reset value. |
| Bits 22:20 | **xxxIM**: Peripheral xxx interrupt mask to CPU2 |
| 0: Peripheral xxx interrupt forwarded to CPU2 |
| 1: Peripheral xxx interrupt to CPU2 masked. |
| Note that bits 22:21 are reserved in STM32WB35xx devices. |
| Bit 19 | Reserved, must be kept at reset value. |
| Bit 18 | **xxxIM**: Peripheral xxx interrupt mask to CPU2 |
| 0: Peripheral xxx interrupt forwarded to CPU2 |
| 1: Peripheral xxx interrupt to CPU2 masked. |
| Bit 17 | Reserved, must be kept at reset value. |
| Bits 16:8 | **xxxIM**: Peripheral xxx interrupt mask to CPU2 |
| 0: Peripheral xxx interrupt forwarded to CPU2 |
| 1: Peripheral xxx interrupt to CPU2 masked. |
| Bit 7 | Reserved, must be kept at reset value. |
| Bits 6:0 | **xxxIM**: Peripheral xxx interrupt mask to CPU2 |
| 0: Peripheral xxx interrupt forwarded to CPU2 |
| 1: Peripheral xxx interrupt to CPU2 masked. |

10.2.16 SYSCFG secure IP control register (SYSCFG_SIPCR)

Address offset: 0x110
System reset value: 0x0000 0000

This register provides write access security and can only be written by the CPU2. A write access from the CPU1 is ignored and a bus error is generated. On any read access the register value is returned.
Bits 31:4  Reserved, must be kept at reset value.

Bit 3  **SRNG**: Enable true RNG security
  0: True RNG security disabled (RNG registers can be accessed by both CPU1 and CPU2)
  1: True RNG security enabled (RNG registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value).

Bit 2  **SPKA**: Enable PKA security
  0: PKA security disabled (PKA registers can be accessed by both CPU1 and CPU2)
  1: PKA security enabled. PKA registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value. When this bit is set and a system reset occurs (including a reset due to wakeup from Standby) the PKA RAM is erased.

Bit 1  **SAES2**: Enable AES2 security
  0: AES2 security disabled (AES2 registers can be accessed by both CPU1 and CPU2)
  1: AES2 security enabled. (AES2 registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return 0 value).

Bit 0  **SAES1**: Enable AES1 KEY[7:0] security
  0: AES1 KEY[7:0] security disabled (AES1 KEY[7:0] registers can be accessed by both CPU1 and CPU2)
  1: AES1 KEY[7:0] security enabled (AES1 KEY[7:0] registers can only be accessed by the CPU2. Write accesses by the CPU1 generate a bus error, read access return zero value).
10.2.17 **SYSCFG register map**

The following table summarizes the SYSCFG 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>
<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>
| 0x000  | SYSCFG_   |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | MEM
| MEMMRM |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |     
|        | Reset value| 0  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | SYSCFG_CFGR1| FPU_IE[5:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008  | SYSCFG_EXTICR1|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x00C  | SYSCFG_EXTICR2|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x010  | SYSCFG_EXTICR3|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x014  | SYSCFG_EXTICR4|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x018  | SYSCFG_SCSR|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | SYSCFG_CFGR2|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value| 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | SYSCFG_SWPR|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x024  | SYSCFG_SKR|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x028  | SYSCFG_SWPR2|        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
Table 46. 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  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x100  | SYSCFG_IMR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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  | SYSCFG_IMR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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  | SYSCFG_C2IMR1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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  | SYSCFG_C2IMR2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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  | SYSCFG_SIPCR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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 66 for the register boundary addresses.
11 Direct memory access controller (DMA)

11.1 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 are two instances of DMA, DMA1 and DMA2.

Each channel is dedicated to managing memory access requests from one or more peripherals. Each DMA includes an arbiter for handling the priority between DMA requests.

11.2 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 (four 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.
11.3 DMA implementation

11.3.1 DMA1 and DMA2

DMA1 and DMA2 are implemented with the hardware configuration parameters shown in the table below.

<table>
<thead>
<tr>
<th>Feature</th>
<th>DMA1</th>
<th>DMA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of channels</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>
11.3.2 DMA request mapping

11.4 DMA functional description

11.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.
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.

11.4.2 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.

11.4.3 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 an 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.

11.4.4 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 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.

**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 the 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 cannot be set again by software to reactivate 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/enableing 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):

\[
\text{DIR} = 1 \text{ defines typically a memory-to-peripheral transfer. More generally, if } \text{DIR} = 1: \\
\quad \text{The source attributes are defined by the DMA_MARx register, the MSIZE}[1:0] \text{ field, and MINC bit of the DMA_CCRx register.} \\
\quad \text{Regardless of their usual naming, these 'memory' register, field, and bit are used to define the source peripheral in peripheral-to-peripheral mode.} \\
\quad \text{The destination attributes are defined by the DMA_PARx register, the PSIZE}[1:0] \text{ field and the PINC bit of the DMA_CCRx register.} \\
\quad \text{Regardless of their usual naming, these 'peripheral' register, field, and bit are used to define the destination memory in memory-to-memory mode.}
\]

\[
\text{DIR} = 0 \text{ defines typically a peripheral-to-memory transfer. More generally, if } \text{DIR} = 0: \\
\quad \text{The source attributes are defined by the DMA_PARx register, the PSIZE}[1:0] \text{ field and the PINC bit of the DMA_CCRx register.} \\
\quad \text{Regardless of their usual naming, these 'peripheral' register, field, and bit are used to define the source memory in memory-to-memory mode.} \\
\quad \text{The destination attributes are defined by the DMA_MARx register, the MSIZE}[1:0] \text{ field and the MINC bit of the DMA_CCRx register.} \\
\quad \text{Regardless of their usual naming, these 'memory' register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.}
\]
11.4.5 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 the table below.

Table 48. 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>Destination 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 8 4</td>
<td></td>
<td></td>
<td>@0x0 / 0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3</td>
<td>1: read B0[7:0] @0x0 then write B0[7:0] @0x0</td>
<td>@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: read B1[7:0] @0x1 then write B1[7:0] @0x1</td>
<td>@0x0 / 0xB0 @0x2 / 0x1B @0x4 / 0x0B2 @0x6 / 0x0B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: read B2[7:0] @0x2 then write B2[7:0] @0x2</td>
<td>@0x0 / 0x0B0 @0x2 / 0x0B1 @0x4 / 0x00B2 @0x6 / 0x0B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: read B3[7:0] @0x3 then write B3[7:0] @0x3</td>
<td>@0x0 / 0x000B0 @0x4 / 0x000B1 @0x8 / 0x000B2 @0xC / 0x0000B3</td>
</tr>
<tr>
<td>16 8 4</td>
<td></td>
<td></td>
<td>@0x0 / 0x1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
<td>1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0</td>
<td>@0x0 / B0 @0x1 / B2 @0x4 / B4 @0x6 / B7B6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1</td>
<td>@0x0 / 0x1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2</td>
<td>@0x0 / 0x1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3</td>
<td>@0x0 / 0x1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
</tr>
<tr>
<td>16 32 4</td>
<td></td>
<td></td>
<td>@0x0 / 0x1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6</td>
<td>1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0</td>
<td>@0x0 / 0x000B1B0 @0x2 / 0x000B2B2 @0x8 / 0x000B4B4 @0xC / 0x0000B7B6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: read B3B2[15:0] @0x2 then write 00B2[15:0] @0x2</td>
<td>@0x0 / 0x000B1B0 @0x2 / 0x000B2B2 @0x8 / 0x000B4B4 @0xC / 0x0000B7B6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: read B5B4[15:0] @0x4 then write 00B4[15:0] @0x4</td>
<td>@0x0 / 0x000B1B0 @0x2 / 0x000B2B2 @0x8 / 0x000B4B4 @0xC / 0x0000B7B6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0x6</td>
<td>@0x0 / 0x000B1B0 @0x2 / 0x000B2B2 @0x8 / 0x000B4B4 @0xC / 0x0000B7B6</td>
</tr>
<tr>
<td>32 8 4</td>
<td></td>
<td></td>
<td>@0x0 / 0x3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8B @0xC / BFBEBD8B</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x4</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: read BBBAB9B8B[31:0] @0x8 then write B8[7:0] @0x2</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: read BFBEBD8B[31:0] @0xC then write BFBE8B[15:0] @0x6</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td>32 16 4</td>
<td></td>
<td></td>
<td>@0x0 / 0x3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8B @0xC / BFBEBD8B</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B1B0[15:0] @0x0</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: read B7B6B5B4[31:0] @0x4 then write B5B4[15:0] @0x2</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: read BBBAB9B8B[31:0] @0x8 then write BBBAB9B8B[31:0] @0x8</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: read BFBEBD8B[31:0] @0xC then write BFBE8B[15:0] @0x6</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td>32 32 4</td>
<td></td>
<td></td>
<td>@0x0 / 0x3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8B @0xC / BFBEBD8B</td>
<td>1: read B3B2B1B0[31:0] @0x0 then write B3B2B1B0[31:0] @0x0</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: read B7B6B5B4[31:0] @0x4 then write B7B6B5B4[31:0] @0x4</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: read BBBAB9B8B[31:0] @0x8 then write BBBAB9B8B[31:0] @0x8</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: read BFBEBD8B[31:0] @0xC then write BFBE8B[31:0] @0xC</td>
<td>@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC</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 the 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.

11.4.6 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 cannot be set again by software (channel x reactivated) 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.
11.5 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 49. 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>

11.6 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).

11.6.1 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.

<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>TEIF4</td>
<td>HTIF4</td>
<td>TCIF4</td>
<td>GIF4</td>
<td>TEIF3</td>
<td>HTIF3</td>
<td>TCIF3</td>
<td>GIF3</td>
<td>TEIF2</td>
<td>HTIF2</td>
<td>TCIF2</td>
<td>GIF2</td>
<td>TEIF1</td>
<td>HTIF1</td>
<td>TCIF1</td>
<td>GIF1</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>
</tbody>
</table>

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: An 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: An 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: An 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: An 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: An 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: An 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: An 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.
### 11.6.2 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>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>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>CTEIF7</td>
<td>CHTIF7</td>
<td>CTCIF7</td>
<td>CGIF7</td>
<td>CTEIF6</td>
<td>CHTIF6</td>
<td>CTCIF6</td>
<td>CGIF6</td>
<td>CTEIF5</td>
<td>CHTIF5</td>
<td>CTCIF5</td>
<td>CGIF5</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>
<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>CTEIF4</td>
<td>CHTIF4</td>
<td>CTCIF4</td>
<td>CGIF4</td>
<td>CTEIF3</td>
<td>CHTIF3</td>
<td>CTCIF3</td>
<td>CGIF3</td>
<td>CTEIF2</td>
<td>CHTIF2</td>
<td>CTCIF2</td>
<td>CGIF2</td>
<td>CTEIF1</td>
<td>CHTIF1</td>
<td>CTCIF1</td>
<td>CGIF1</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>

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
11.6.3 **DMA channel x configuration register (DMA_CCRx)**

Address offset: \(0x08 + 0x14 \times (x - 1)\), \((x = 1 \text{ to } 7)\)

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>Bit 31:15 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 14 MEM2MEM: Memory-to-memory mode</td>
</tr>
<tr>
<td>0: Disabled</td>
</tr>
<tr>
<td>1: Enabled</td>
</tr>
<tr>
<td><strong>Note:</strong> 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).</td>
</tr>
<tr>
<td>Bit 13:12 PL[1:0]: Priority level</td>
</tr>
<tr>
<td>00: Low</td>
</tr>
<tr>
<td>01: Medium</td>
</tr>
<tr>
<td>10: High</td>
</tr>
<tr>
<td>11: Very high</td>
</tr>
<tr>
<td><strong>Note:</strong> This bitfield 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).</td>
</tr>
</tbody>
</table>
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 bitfield identifies the memory source if DIR = 1 and the memory destination if DIR = 0.
In peripheral-to-peripheral mode, this bitfield 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 bitfield 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 bitfield identifies the memory destination if DIR = 1 and the memory source if DIR = 0.
In peripheral-to-peripheral mode, this bitfield 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 bitfield 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 bit identifies the memory source if DIR = 1 and the memory destination if DIR = 0.
In peripheral-to-peripheral mode, this bit 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 bit identifies the memory destination if DIR = 1 and the memory source if DIR = 0.
In peripheral-to-peripheral mode, this bit 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

- **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

- **0**: Disabled
- **1**: Enabled

*Note:* This bit is set and cleared by software.

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel 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
### 11.6.4 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

<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></td>
<td></td>
<td></td>
<td></td>
<td></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>

**NDT[15:0]**

- Bits 31:16: Reserved, must be kept at reset value.
- Bits 15:0 **NDT[15:0]:** Number of data to transfer (0 to 2^16 - 1)
  - This bitfield 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 bitfield is zero, no transfer can be served whatever the channel status (enabled or not).

  **Note:** This bitfield 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).

### 11.6.5 DMA channel x peripheral address register (DMA_CPARx)

Address offset: 0x10 + 0x14 * (x - 1), (x = 1 to 7)

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></td>
<td></td>
<td></td>
<td></td>
<td></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>

**PA[31:16]**

**PA[15:0]**

- Bits 31:16: Reserved, must be kept at reset value.
- Bits 15:0 **PA[15:0]:** Peripheral address
Bits 31:0  **PA[31:0]**: Peripheral address

It contains the base address of the peripheral data register from/to which the data is 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[1:0] = 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 bitfield identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral destination address if DIR = 1 and the peripheral source address if DIR = 0.

**Note:** This bitfield 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).

### 11.6.6 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>MA[31:16]</th>
</tr>
</thead>
<tbody>
<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>15    14    13    12    11    10     9     8     7     6     5     4     3     2     1     0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MA[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **MA[31:0]**: Peripheral address

It contains the base address of the memory from/to which the data is read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE[1:0] = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this bitfield identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral source address if DIR = 1 and the peripheral destination address if DIR = 0.

**Note:** This bitfield 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).

### 11.6.7 DMA register map

#### Table 50. DMA register map and reset values

<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>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DMA_ISR</td>
<td>CTB7 CTB6 CTB5 CTB4 CTB3 CTB2 CTB1 CTF7 CTF6 CTF5 CTF4 CTF3 CTF2 CTF1 CTF0 GTK7 GTK6 GTK5 GTK4 GTK3 GTK2 GTK1 GTK0 HTK7 HTK6 HTK5 HTK4 HTK3 HTK2 HTK1 HTK0 GF7 GF6 GF5 GF4 GF3 GF2 GF1 GF0</td>
</tr>
<tr>
<td></td>
<td>Reset value</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>
</tr>
<tr>
<td>0x004</td>
<td>DMA_IFCR</td>
<td>CTB7 CTB6 CTB5 CTB4 CTB3 CTB2 CTB1 CTF7 CTF6 CTF5 CTF4 CTF3 CTF2 CTF1 CTF0 GTK7 GTK6 GTK5 GTK4 GTK3 GTK2 GTK1 GTK0 HTK7 HTK6 HTK5 HTK4 HTK3 HTK2 HTK1 HTK0 GF7 GF6 GF5 GF4 GF3 GF2 GF1 GF0</td>
</tr>
<tr>
<td></td>
<td>Reset value</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>
</tr>
</tbody>
</table>

346/1530  RM0434 Rev 13
### Table 50. DMA register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x008</td>
<td>DMA_CCR1</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>MEM2MEM: MEM[1:0], PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, CIRC, DIR, TEIE, HTIE, TCIE, EN</td>
</tr>
<tr>
<td>0x00C</td>
<td>DMA_CNDTR1</td>
<td>NDTR[15:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x010</td>
<td>DMA_CPAR1</td>
<td>PA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x014</td>
<td>DMA_CMAR1</td>
<td>MA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x018</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>DMA_CCR2</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>MEM2MEM: MEM[1:0], PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, CIRC, DIR, TEIE, HTIE, TCIE, EN</td>
</tr>
<tr>
<td>0x020</td>
<td>DMA_CNDTR2</td>
<td>NDTR[15:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x024</td>
<td>DMA_CPAR2</td>
<td>PA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x028</td>
<td>DMA_CMAR2</td>
<td>MA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x02C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>DMA_CCR3</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>MEM2MEM: MEM[1:0], PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, CIRC, DIR, TEIE, HTIE, TCIE, EN</td>
</tr>
<tr>
<td>0x034</td>
<td>DMA_CNDTR3</td>
<td>NDTR[15:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x038</td>
<td>DMA_CPAR3</td>
<td>PA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x03C</td>
<td>DMA_CMAR3</td>
<td>MA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x040</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>DMA_CCR4</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>MEM2MEM: MEM[1:0], PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, CIRC, DIR, TEIE, HTIE, TCIE, EN</td>
</tr>
<tr>
<td>0x048</td>
<td>DMA_CNDTR4</td>
<td>NDTR[15:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x04C</td>
<td>DMA_CPAR4</td>
<td>PA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x050</td>
<td>DMA_CMAR4</td>
<td>MA[31:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>0x054</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x058</td>
<td>DMA_CCR5</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>MEM2MEM: MEM[1:0], PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, CIRC, DIR, TEIE, HTIE, TCIE, EN</td>
</tr>
</tbody>
</table>
Table 50. DMA 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  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x05C  | DMA_CNDTR5    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x060  | DMA_CPAR5     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x064  | DMA_CMAR5     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x068  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x06C  | DMA_CCR6      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x070  | DMA_CNDTR6    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x074  | DMA_CPAR6     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x078  | DMA_CMAR6     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x07C  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x080  | DMA_CCR7      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x084  | DMA_CNDTR7    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x088  | DMA_CPAR7     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |
| 0x08C  | DMA_CMAR7     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |    |

Refer to Section 2.2 for the register boundary addresses.
12 DMA request multiplexer (DMAMUX)

12.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

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 12.3.1.

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 upon product implementation. They are detailed in Section 12.3.2.
12.2 DMAMUX main features

- 14-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 20 trigger inputs to DMA request generator
- 20 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:
- 36 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

12.3 DMAMUX implementation

Table 51. DMAMUX implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA request SPI2</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

12.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

Table 52. DMAMUX instantiation

<table>
<thead>
<tr>
<th>Feature</th>
<th>DMAMUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of DMAMUX output request channels</td>
<td>14</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>20</td>
</tr>
<tr>
<td>Number of DMAMUX synchronization inputs</td>
<td>20</td>
</tr>
<tr>
<td>Number of DMAMUX peripheral request inputs</td>
<td>36</td>
</tr>
</tbody>
</table>

12.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.
DMAMUX is used with DMA1 and DMA2:
DMAMUX channels 0 to 6 are connected to DMA1 channels 1 to 7
DMAMUX channels 7 to 13 are connected to DMA2 channels 1 to 7

Table 53. 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_CH2</td>
<td>43</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>dmamux_req_gen1</td>
<td>23</td>
<td>TIM1_CH3</td>
<td>44</td>
<td>Reserved</td>
</tr>
<tr>
<td>3</td>
<td>dmamux_req_gen2</td>
<td>24</td>
<td>TIM1_CH4</td>
<td>45</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>dmamux_req_gen3</td>
<td>25</td>
<td>TIM1_UP</td>
<td>46</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>ADC1</td>
<td>26</td>
<td>TIM1_TRIG</td>
<td>47</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>SPI1_RX</td>
<td>27</td>
<td>TIM1_COM</td>
<td>48</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>SPI1_TX</td>
<td>28</td>
<td>TIM2_CH1</td>
<td>49</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>SPI2_RX</td>
<td>29</td>
<td>TIM2_CH2</td>
<td>50</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>SPI2_TX</td>
<td>30</td>
<td>TIM2_CH3</td>
<td>51</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>I2C1_RX</td>
<td>31</td>
<td>TIM2_CH4</td>
<td>52</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>I2C1_TX</td>
<td>32</td>
<td>TIM2_UP</td>
<td>53</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>I2C3_RX</td>
<td>33</td>
<td>TIM16_CH1</td>
<td>54</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>I2C3_TX</td>
<td>34</td>
<td>TIM16_UP</td>
<td>55</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>USART1_RX</td>
<td>35</td>
<td>TIM17_CH1</td>
<td>56</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>USART1_TX</td>
<td>36</td>
<td>TIM17_UP</td>
<td>57</td>
<td>Reserved</td>
</tr>
<tr>
<td>16</td>
<td>LPUART1_RX</td>
<td>37</td>
<td>AES1_IN</td>
<td>58</td>
<td>Reserved</td>
</tr>
<tr>
<td>17</td>
<td>LPUART1_TX</td>
<td>38</td>
<td>AES1_OUT</td>
<td>59</td>
<td>Reserved</td>
</tr>
<tr>
<td>18</td>
<td>SAI1_A</td>
<td>39</td>
<td>AES2_IN</td>
<td>60</td>
<td>Reserved</td>
</tr>
<tr>
<td>19</td>
<td>SAI1_B</td>
<td>40</td>
<td>AES2_OUT</td>
<td>61</td>
<td>Reserved</td>
</tr>
<tr>
<td>20</td>
<td>QUADSPI</td>
<td>41</td>
<td>Reserved</td>
<td>62</td>
<td>Reserved</td>
</tr>
<tr>
<td>21</td>
<td>TIM1_CH1</td>
<td>42</td>
<td>Reserved</td>
<td>63</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 54. 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>16</td>
<td>dmamux_evt0</td>
</tr>
<tr>
<td>1</td>
<td>EXTI LINE1</td>
<td>17</td>
<td>dmamux_evt1</td>
</tr>
<tr>
<td>2</td>
<td>EXTI LINE2</td>
<td>18</td>
<td>LPTIM1_OUT</td>
</tr>
<tr>
<td>3</td>
<td>EXTI LINE3</td>
<td>19</td>
<td>LPTIM2_OUT</td>
</tr>
<tr>
<td>4</td>
<td>EXTI LINE4</td>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>EXTI LINE5</td>
<td>21</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>EXTI LINE6</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>EXTI LINE7</td>
<td>23</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### Table 54. 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>8</td>
<td>EXTI LINE8</td>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>EXTI LINE9</td>
<td>25</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>EXTI LINE10</td>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>EXTI LINE11</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>EXTI LINE12</td>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>EXTI LINE13</td>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>EXTI LINE14</td>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>EXTI LINE15</td>
<td>31</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Table 55. 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>16</td>
<td>dmamux_evt0</td>
</tr>
<tr>
<td>1</td>
<td>EXTI LINE1</td>
<td>17</td>
<td>dmamux_evt1</td>
</tr>
<tr>
<td>2</td>
<td>EXTI LINE2</td>
<td>18</td>
<td>LPTIM1_OUT</td>
</tr>
<tr>
<td>3</td>
<td>EXTI LINE3</td>
<td>19</td>
<td>LPTIM2_OUT</td>
</tr>
<tr>
<td>4</td>
<td>EXTI LINE4</td>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>EXTI LINE5</td>
<td>21</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>EXTI LINE6</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>EXTI LINE7</td>
<td>23</td>
<td>Reserved</td>
</tr>
<tr>
<td>8</td>
<td>EXTI LINE8</td>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>EXTI LINE9</td>
<td>25</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>EXTI LINE10</td>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>EXTI LINE11</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>EXTI LINE12</td>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>EXTI LINE13</td>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>EXTI LINE14</td>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>EXTI LINE15</td>
<td>31</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
12.4 DMAMUX functional description

12.4.1 DMAMUX block diagram

*Figure 26* shows the DMAMUX block diagram.

**Figure 26. 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)
12.4.2 DMAMUX signals

*Table 56* 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_rqxx</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>

12.4.3 DMAMUX channels

A DMAMUX channel is a request multiplexer channel that can include, depending upon the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to a single channel of DMA controller(s).

**Channel configuration procedure**

Follow the sequence below to configure 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.

12.4.4 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.
Caution: A same non-null DMAREQ_ID cannot be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX_CxCR and DMAMUX_CyCR), except when the application guarantees that the two connected DMA channels are not simultaneously active.

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 a programmable rising/falling edge is detected on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, internally to the DMAMUX request multiplexer, there is a programmable DMA request counter, which can be used for the channel request output generation, and 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 28*, 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 is discarded. The following asserted input request lines is not 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 deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in the 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 the NBREQ field, plus one.

**Note:** The NBREQ field value can be written by software only when both synchronization enable bit (SE) and event generation enable bit (EGE) 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 27 and Figure 28.
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 register.

Note: The request multiplexer channel x synchronization must be disabled (DMAMUX_CxCR.SE = 0) when the use of the related channel of the DMA controller is completed. Else, upon a new detected synchronization event, there is 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.

12.4.5 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 deasserted), 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 is 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 can be written by software only when the enable GE bit of the corresponding generator channel x is disabled.
There is no hardware write protection.
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 DMAMUX_RGxSR register.

Note: The request generator channel x must be disabled (DMAMUX_RGxCR.GE = 0) when the usage of the related channel of the DMA controller is completed. Else, upon a new detected trigger event, there is 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.

12.5 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 57. 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>
12.6 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX base address. DMAMUX registers may be accessed per byte (8-bit), half-word (16-bit), or word (32-bit). The address must be aligned with the data size.

12.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x0000 + 0x04 * x (x = 0 to 13)

Reset value: 0x0000 0000

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 55: 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 must 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 (no synchronization, no detection).
01: Rising edge
10: Falling edge
11: Rising and falling edges

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
12.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)

Address offset: 0x080
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>Res.</td>
<td>Res.</td>
<td>SOF13</td>
<td>SOF12</td>
<td>SOF11</td>
<td>SOF10</td>
<td>SOF9</td>
<td>SOF8</td>
<td>SOF7</td>
<td>SOF6</td>
<td>SOF5</td>
<td>SOF4</td>
<td>SOF3</td>
<td>SOF2</td>
<td>SOF1</td>
<td>SOF0</td>
</tr>
<tr>
<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>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 5:0 **DMAREQ_ID[5:0]:** DMA request identification
Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

12.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)

Address offset: 0x084
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>Res.</td>
<td>Res.</td>
<td>CSOF13</td>
<td>CSOF12</td>
<td>CSOF11</td>
<td>CSOF10</td>
<td>CSOF9</td>
<td>CSOF8</td>
<td>CSOF7</td>
<td>CSOF6</td>
<td>CSOF5</td>
<td>CSOF4</td>
<td>CSOF3</td>
<td>CSOF2</td>
<td>CSOF1</td>
<td>CSOF0</td>
</tr>
<tr>
<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>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **CSOF[13:0]:** 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.


12.6.4       DMAMUX request generator channel x configuration register
(DMAMUX_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 3)

Reset value: 0x0000 0000

| Bit 31:24 | Reserved, must be kept at reset value. |
| Bit 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 must be written only when GE bit is disabled. |
| Bit 18:17 | **GPOLO[1:0]**: DMA request generator trigger polarity |
|           | Defines the edge polarity of the selected trigger input |
|           | 00: No event, i.e. no trigger detection nor generation. |
|           | 01: Rising edge |
|           | 10: Falling edge |
|           | 11: Rising and falling edges |
| Bit 16    | **GE**: DMA request generator channel x enable |
|           | 0: DMA request generator channel x disabled |
|           | 1: DMA request generator channel x enabled |
| Bit 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 |
| Bit 7:5   | Reserved, must be kept at reset value. |
| Bit 4:0   | **SIG_ID[4:0]**: Signal identification |
|           | Selects the DMA request trigger input used for the channel x of the DMA request generator |
### 12.6.5 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_RGxCR register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

### 12.6.6 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>
</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.
### 12.6.7 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>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>28</th>
<th>25</th>
<th>22</th>
<th>19</th>
<th>16</th>
<th>13</th>
<th>10</th>
<th>7</th>
<th>4</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DMAMUX_C0CR</td>
<td></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>
</tr>
<tr>
<td>0x004</td>
<td>DMAMUX_C1CR</td>
<td></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>
</tr>
<tr>
<td>0x008</td>
<td>DMAMUX_C2CR</td>
<td></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>
</tr>
<tr>
<td>0x00C</td>
<td>DMAMUX_C3CR</td>
<td></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>
</tr>
<tr>
<td>0x010</td>
<td>DMAMUX_C4CR</td>
<td></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>
</tr>
<tr>
<td>0x014</td>
<td>DMAMUX_C5CR</td>
<td></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>
</tr>
<tr>
<td>0x018</td>
<td>DMAMUX_C6CR</td>
<td></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>
</tr>
<tr>
<td>0x01C</td>
<td>DMAMUX_C7CR</td>
<td></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>
</tr>
<tr>
<td>0x020</td>
<td>DMAMUX_C8CR</td>
<td></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>
</tr>
<tr>
<td>0x024</td>
<td>DMAMUX_C9CR</td>
<td></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>
</tr>
<tr>
<td>0x028</td>
<td>DMAMUX_C10CR</td>
<td></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>
</tr>
<tr>
<td>0x02C</td>
<td>DMAMUX_C11CR</td>
<td></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>
</tr>
<tr>
<td>0x030</td>
<td>DMAMUX_C12CR</td>
<td></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>
</tr>
<tr>
<td>0x034</td>
<td>DMAMUX_C13CR</td>
<td></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>
</tr>
</tbody>
</table>

**Table 58. DMAMUX register map and reset values**
Refer to Section 2.2 on page 66 for the register boundary addresses.
13 Nested vectored interrupt controller (NVIC)

13.1 NVIC main features

The CPU1 NVIC features:
- 63 maskable interrupt channels (not including the sixteen Cortex®-M4 with FPU interrupt lines)
- 16 programmable priority levels (four bits of interrupt priority are used)
- Low-latency exception interrupt handling
- Power management control
- Implementation of System control registers

The CPU2 NVIC features:
- 32 maskable interrupt channels (not including the sixteen Cortex®-M0+ interrupt lines)
- Four programmable priority levels (two bits of interrupt priority are used)
- Low-latency exception interrupt handling
- Power management control

The NVICs and the processor cores interfaces are closely coupled, resulting in 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 STM32 Cortex®-M4 MCUs and MPUs programming manual (PM0214) for Cortex®-M4, and Cortex®-M0+ programming manual for STM32L0, STM32G0, STM32WL and STM32WB Series (PM0223) for Cortex®-M0+.

13.2 NVIC implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI2 interrupt</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>TSC interrupt</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LCD interrupt</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LCD wakeup</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

13.3 Interrupt block diagram

The different peripheral interrupts are connected in different ways, depending on the sharing between the two CPUs. To prevent a peripheral or EXTI interrupt to trigger both CPUs, they can be masked either in the NVIC, or, for the NVIC vector sharing multiple peripheral interrupts, by a pre-mask in the SYSCFG registers, see Section 10: System configuration controller (SYSCFG).

The interrupt block diagram is shown in Figure 29.
13.4 **Interrupt and exception vectors**

Each of the CPU1 and CPU2 has its own vector table, see, respectively, *Table 60* and *Table 61*, where shaded cells indicate the processor exceptions.
### Table 60. CPU1 vector table

<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></td>
<td>0x0000 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td>0x0000 0000</td>
</tr>
<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</td>
<td>Non maskable interrupt HSE CSS, Flash ECC, and SRAM2 parity</td>
<td>0x0000 0008</td>
</tr>
<tr>
<td></td>
<td>- -1</td>
<td>Fixed</td>
<td>HardFault</td>
<td>All classes of fault</td>
<td>0x0000 000C</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Settable</td>
<td>MemManager</td>
<td>Memory manager</td>
<td>0x0000 0010</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Settable</td>
<td>BusFault</td>
<td>Pre-fetch fault, memory access fault</td>
<td>0x0000 0014</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Settable</td>
<td>UsageFault</td>
<td>Undefined instruction or illegal state</td>
<td>0x0000 0018</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td>0x0000 001C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td>0x0000 0028</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Settable</td>
<td>SVCall</td>
<td>System service call via SWI instruction</td>
<td>0x0000 002C</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Settable</td>
<td>Debug</td>
<td>Debug monitor</td>
<td>0x0000 0030</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td>0x0000 0034</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Settable</td>
<td>PendSV</td>
<td>Pendable request for system service</td>
<td>0x0000 0038</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Settable</td>
<td>Systick</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 early wakeup</td>
<td>0x0000 0040</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>Settable</td>
<td>PVD, PVM1, PVM3</td>
<td>PVD through EXTI[16] (C1IMR2[20])</td>
<td>0x0000 0044</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PVM1 through EXTI[31] (C1IMR2[16])</td>
<td>0x0000 0044</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PVM3 through EXTI[33] (C1IMR2[18])</td>
<td>0x0000 0044</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>Settable</td>
<td>TAMP, RTC_STAMP, LSE_CSS</td>
<td>Tamper, TimeStamp, LSECSS interrupt through EXTI[18]</td>
<td>0x0000 0048</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>Settable</td>
<td>RTC_WKUP</td>
<td>RTC wakeup interrupt through EXTI[19]</td>
<td>0x0000 004C</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>Settable</td>
<td>Flash</td>
<td>Flash memory global interrupt and Flash memory ECC single error interrupt</td>
<td>0x0000 0050</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>Settable</td>
<td>RCC</td>
<td>RCC global interrupt</td>
<td>0x0000 0054</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>Settable</td>
<td>EXTI0</td>
<td>EXTI line 0 interrupt through EXTI[0]</td>
<td>0x0000 0058</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>Settable</td>
<td>EXTI1</td>
<td>EXTI line 1 interrupt through EXTI[1]</td>
<td>0x0000 005C</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>Settable</td>
<td>EXTI2</td>
<td>EXTI line 2 interrupt through EXTI[2]</td>
<td>0x0000 0060</td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>Settable</td>
<td>EXTI3</td>
<td>EXTI line 3 interrupt through EXTI[3]</td>
<td>0x0000 0064</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>Settable</td>
<td>EXTI4</td>
<td>EXTI line 4 interrupt through EXTI[4]</td>
<td>0x0000 0068</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>Settable</td>
<td>DMA1_CH1</td>
<td>DMA1 channel 1 interrupt</td>
<td>0x0000 006C</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
<td>Settable</td>
<td>DMA1_CH2</td>
<td>DMA1 channel 2 interrupt</td>
<td>0x0000 0070</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>Settable</td>
<td>DMA1_CH3</td>
<td>DMA1 channel 3 interrupt</td>
<td>0x0000 0074</td>
</tr>
<tr>
<td>14</td>
<td>21</td>
<td>Settable</td>
<td>DMA1_CH4</td>
<td>DMA1 channel 4 interrupt</td>
<td>0x0000 0078</td>
</tr>
<tr>
<td>15</td>
<td>22</td>
<td>Settable</td>
<td>DMA1_CH5</td>
<td>DMA1 channel 5 interrupt</td>
<td>0x0000 007C</td>
</tr>
<tr>
<td>Position</td>
<td>Priority</td>
<td>Type of priority</td>
<td>Acronym</td>
<td>Description</td>
<td>Address</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>16</td>
<td>23</td>
<td>Settable</td>
<td>DMA1_CH6</td>
<td>DMA1 channel 6 interrupt</td>
<td>0x0000 0080</td>
</tr>
<tr>
<td>17</td>
<td>24</td>
<td>Settable</td>
<td>DMA1_CH7</td>
<td>DMA1 channel 7 interrupt</td>
<td>0x0000 0084</td>
</tr>
<tr>
<td>18</td>
<td>25</td>
<td>Settable</td>
<td>ADC1</td>
<td>ADC1 global interrupt</td>
<td>0x0000 0088</td>
</tr>
<tr>
<td>19</td>
<td>26</td>
<td>Settable</td>
<td>USB_HP</td>
<td>USB high priority interrupt</td>
<td>0x0000 008C</td>
</tr>
<tr>
<td>20</td>
<td>27</td>
<td>Settable</td>
<td>USB_LP</td>
<td>USB low priority interrupt (including USB wakeup)</td>
<td>0x0000 0090</td>
</tr>
<tr>
<td>21</td>
<td>28</td>
<td>Settable</td>
<td>C2SEV PWR_C2H</td>
<td>CPU2 SEV through EXTI[40] PWR CPU2 HOLD wakeup interrupt</td>
<td>0x0000 0094</td>
</tr>
<tr>
<td>22</td>
<td>29</td>
<td>Settable</td>
<td>COMP</td>
<td>COMP2 and COMP1 interrupt through EXTI[21:20]</td>
<td>0x0000 0098</td>
</tr>
<tr>
<td>24</td>
<td>31</td>
<td>Settable</td>
<td>TIM1_BRK</td>
<td>Timer 1 break interrupt</td>
<td>0x0000 00A0</td>
</tr>
<tr>
<td>25</td>
<td>32</td>
<td>Settable</td>
<td>TIM1_UP TIM16</td>
<td>Timer 1 update (C1IMR1[13]) Timer 16 global interrupt (C1IMR1[14])</td>
<td>0x0000 00A4</td>
</tr>
<tr>
<td>26</td>
<td>33</td>
<td>Settable</td>
<td>TIM1_TRG_COM TIM17</td>
<td>Timer 1 trigger and communication (C1IMR1[13]) Timer 17 global interrupt (C1IMR1[15])</td>
<td>0x0000 00A8</td>
</tr>
<tr>
<td>27</td>
<td>34</td>
<td>Settable</td>
<td>TIM1_CC</td>
<td>Timer 1 capture compare interrupt</td>
<td>0x0000 00AC</td>
</tr>
<tr>
<td>28</td>
<td>35</td>
<td>Settable</td>
<td>TIM2</td>
<td>Timer 2 global interrupt</td>
<td>0x0000 00B0</td>
</tr>
<tr>
<td>29</td>
<td>36</td>
<td>Settable</td>
<td>PKA</td>
<td>Private key accelerator interrupt</td>
<td>0x0000 00B4</td>
</tr>
<tr>
<td>30</td>
<td>37</td>
<td>Settable</td>
<td>I2C1_EV</td>
<td>I2C1 event interrupt</td>
<td>0x0000 00B8</td>
</tr>
<tr>
<td>31</td>
<td>38</td>
<td>Settable</td>
<td>I2C1_ER</td>
<td>I2C1 error interrupt</td>
<td>0x0000 00BC</td>
</tr>
<tr>
<td>32</td>
<td>39</td>
<td>Settable</td>
<td>I2C3_EV</td>
<td>I2C3 event interrupt</td>
<td>0x0000 00C0</td>
</tr>
<tr>
<td>33</td>
<td>40</td>
<td>Settable</td>
<td>I2C3_ER</td>
<td>I2C3 error interrupt</td>
<td>0x0000 00C4</td>
</tr>
<tr>
<td>34</td>
<td>41</td>
<td>Settable</td>
<td>SPI1</td>
<td>SPI 1 global interrupt</td>
<td>0x0000 00C8</td>
</tr>
<tr>
<td>35</td>
<td>42</td>
<td>Settable</td>
<td>SPI2</td>
<td>SPI 2 global interrupt</td>
<td>0x0000 00CC</td>
</tr>
<tr>
<td>36</td>
<td>43</td>
<td>Settable</td>
<td>USART1</td>
<td>USART1 global interrupt</td>
<td>0x0000 00D0</td>
</tr>
<tr>
<td>37</td>
<td>44</td>
<td>Settable</td>
<td>LPUART1</td>
<td>LPUART1 global interrupt</td>
<td>0x0000 00D4</td>
</tr>
<tr>
<td>38</td>
<td>45</td>
<td>Settable</td>
<td>SAI1</td>
<td>SAI1 A and B global interrupt</td>
<td>0x0000 00D8</td>
</tr>
<tr>
<td>39</td>
<td>46</td>
<td>Settable</td>
<td>TSC</td>
<td>TSC global interrupt</td>
<td>0x0000 00DC</td>
</tr>
<tr>
<td>40</td>
<td>47</td>
<td>Settable</td>
<td>EXTI[15:10]</td>
<td>EXTI line [15:10] interrupt through EXTI[15:10] (C1IMR1[31:26])</td>
<td>0x0000 00E0</td>
</tr>
<tr>
<td>41</td>
<td>48</td>
<td>Settable</td>
<td>RTC_ALARM</td>
<td>RTC alarms (A and B) interrupt through EXTI[17]</td>
<td>0x0000 00E4</td>
</tr>
<tr>
<td>42</td>
<td>49</td>
<td>Settable</td>
<td>CRS_IT</td>
<td>CRS interrupt</td>
<td>0x0000 00E8</td>
</tr>
</tbody>
</table>
Table 60. CPU1 vector table (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>43</td>
<td>50</td>
<td>Settable</td>
<td>PWR_SOTF</td>
<td>PWR switching on the fly interrupt</td>
<td>0x0000 00EC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWR_BLEACT</td>
<td>PWR end of BLE activity interrupt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWR_802ACT</td>
<td>PWR end of 802.15.4 activity interrupt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWR_RFPHASE</td>
<td>PWR end of critical radio phase interrupt</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>51</td>
<td>Settable</td>
<td>IPCC_C1_RX_IT</td>
<td>IPCC CPU1 RX occupied interrupt</td>
<td>0x0000 00F0</td>
</tr>
<tr>
<td>45</td>
<td>52</td>
<td>Settable</td>
<td>IPCC_C1_TX_IT</td>
<td>IPCC CPU1 TX free interrupt</td>
<td>0x0000 00F4</td>
</tr>
<tr>
<td>46</td>
<td>53</td>
<td>Settable</td>
<td>HSEM</td>
<td>Semaphore interrupt 0 to CPU1</td>
<td>0x0000 00F8</td>
</tr>
<tr>
<td>47</td>
<td>54</td>
<td>Settable</td>
<td>LPTIM1</td>
<td>LPtimer 1 global interrupt</td>
<td>0x0000 00FC</td>
</tr>
<tr>
<td>48</td>
<td>55</td>
<td>Settable</td>
<td>LPTIM2</td>
<td>LPtimer 2 global interrupt</td>
<td>0x0000 0100</td>
</tr>
<tr>
<td>49</td>
<td>56</td>
<td>Settable</td>
<td>LCD</td>
<td>LCD global interrupt</td>
<td>0x0000 0104</td>
</tr>
<tr>
<td>50</td>
<td>57</td>
<td>Settable</td>
<td>QUADSPI</td>
<td>QUADSPI global interrupt</td>
<td>0x0000 0108</td>
</tr>
<tr>
<td>51</td>
<td>58</td>
<td>Settable</td>
<td>AES1</td>
<td>AES1 global interrupt</td>
<td>0x0000 010C</td>
</tr>
<tr>
<td>52</td>
<td>59</td>
<td>Settable</td>
<td>AES2</td>
<td>AES2 global interrupt</td>
<td>0x0000 0110</td>
</tr>
<tr>
<td>53</td>
<td>60</td>
<td>Settable</td>
<td>True RNG</td>
<td>True random number generator interrupt</td>
<td>0x0000 0114</td>
</tr>
<tr>
<td>54</td>
<td>61</td>
<td>Settable</td>
<td>FPU</td>
<td>Floating point unit interrupt</td>
<td>0x0000 0118</td>
</tr>
<tr>
<td>55</td>
<td>62</td>
<td>Settable</td>
<td>DMA2_CH1</td>
<td>DMA2_channel 1 interrupt</td>
<td>0x0000 011C</td>
</tr>
<tr>
<td>56</td>
<td>63</td>
<td>Settable</td>
<td>DMA2_CH2</td>
<td>DMA2_channel 2 interrupt</td>
<td>0x0000 0120</td>
</tr>
<tr>
<td>57</td>
<td>64</td>
<td>Settable</td>
<td>DMA2_CH3</td>
<td>DMA2_channel 3 interrupt</td>
<td>0x0000 0124</td>
</tr>
<tr>
<td>58</td>
<td>65</td>
<td>Settable</td>
<td>DMA2_CH4</td>
<td>DMA2_channel 4 interrupt</td>
<td>0x0000 0128</td>
</tr>
<tr>
<td>59</td>
<td>66</td>
<td>Settable</td>
<td>DMA2_CH5</td>
<td>DMA2_channel 5 interrupt</td>
<td>0x0000 012C</td>
</tr>
<tr>
<td>60</td>
<td>67</td>
<td>Settable</td>
<td>DMA2_CH6</td>
<td>DMA2_channel 6 interrupt</td>
<td>0x0000 0130</td>
</tr>
<tr>
<td>61</td>
<td>68</td>
<td>Settable</td>
<td>DMA2_CH7</td>
<td>DMA2_channel 7 interrupt</td>
<td>0x0000 0134</td>
</tr>
<tr>
<td>62</td>
<td>69</td>
<td>Settable</td>
<td>DMAMUX1_OVR</td>
<td>DMAMUX1 overrun interrupt</td>
<td>0x0000 0138</td>
</tr>
</tbody>
</table>
### Table 61. CPU2 vector table

<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>0x0000 0000</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0004</td>
<td></td>
</tr>
<tr>
<td>-14</td>
<td>-3</td>
<td>Fixed</td>
<td>NMI</td>
<td>Non maskable interrupt HSE CSS, Flash ECC, and SRAM2 parity</td>
<td>0x0000 0008</td>
</tr>
<tr>
<td>-13</td>
<td>-1</td>
<td>Fixed</td>
<td>HardFault</td>
<td>All classes of fault</td>
<td>0x0000 000C</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0010</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0028</td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td>0</td>
<td>Settable</td>
<td>SVCall</td>
<td>System service call via SWI instruction</td>
<td>0x0000 002C</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0030</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0034</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>1</td>
<td>Settable</td>
<td>PendSV</td>
<td>Pendable request for system service</td>
<td>0x0000 0038</td>
</tr>
<tr>
<td>-1</td>
<td>2</td>
<td>Settable</td>
<td>Systick</td>
<td>System tick timer</td>
<td>0x0000 003C</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>Settable</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0040</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>Settable</td>
<td>PVD, PVM1, PVM3</td>
<td>PVD through EXTI[16] (C2IMR2[20]) PVM1 through EXTI[31] (C2IMR2[16]) PVM3 through EXTI[33] (C2IMR2[18])</td>
<td>0x0000 0044</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>Settable</td>
<td>RTC_WKUP, TAMPP,RTC_STAMP LSE_CSS,RTC_ALARM</td>
<td>RTC wakeup interrupt through EXTI[19] (C2IMR1[3]) Tamper, TimeStamp LSECSS interrupt through EXTI[18] (C2IMR1[0]) RTC alarms (A and B) interrupt through EXTI[17] (C2IMR1[4])</td>
<td>0x0000 0048</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>Settable</td>
<td>USB_HP, USB_LP CRS_IT</td>
<td>USB high priority interrupt, USB low priority interrupt (including USB wakeup) CRS interrupt</td>
<td>0x0000 004C</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Settable</td>
<td>RCC, FLASH C1SEV</td>
<td>RCC global interrupt (C2IMR1[5]) Flash memory global interrupt and Flash memory ECC single error interrupt (C2IMR1[8]) CPU1 SEV through EXTI[41]</td>
<td>0x0000 0050</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>Settable</td>
<td>EXTI[1:0]</td>
<td>EXTI line 1:0 interrupt through EXTI[1:0] (C2IMR1[17:16])</td>
<td>0x0000 0054</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>Settable</td>
<td>EXTI[3:2]</td>
<td>EXTI line 3:2 interrupt through EXTI[3:2] (C2IMR1[19:18])</td>
<td>0x0000 0058</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>Settable</td>
<td>EXTI[15:4]</td>
<td>EXTI line 15:4 interrupt through EXTI[15:4] (C2IMR1[31:20])</td>
<td>0x0000 005C</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>Settable</td>
<td>TSC 802_IT0</td>
<td>TSC global interrupt (C2IMR2[21]) 802.15.4 interrupt 0</td>
<td>0x0000 0060</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>Settable</td>
<td>DMA1_CH[3:1]</td>
<td>DMA1 channel 3:1 interrupt (C2IMR2[2.0])</td>
<td>0x0000 0064</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>Settable</td>
<td>DMA1_CH[7:4]</td>
<td>DMA1 channel 7:4 interrupt (C2IMR2[6.3])</td>
<td>0x0000 0068</td>
</tr>
</tbody>
</table>
Table 61. CPU2 vector table (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>11</td>
<td>14</td>
<td>Settable</td>
<td>DMA2_CH[7:1]</td>
<td>DMA2 channel 7-1 interrupt (C2IMR2[14:8])</td>
<td>0x0000 006C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DMAMUX1_OVR</td>
<td>DMAMUX1 overrun interrupt (C2IMR2[15])</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>Settable</td>
<td>ADC1</td>
<td>ADC1 global interrupt (C2IMR1[12])</td>
<td>0x0000 0070</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>COMP</td>
<td>COMP1 and COMP2 interrupt through EXTI[21:20] (C2IMR1[11])</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>Settable</td>
<td>LPTIM1</td>
<td>LP timer 1 global interrupt</td>
<td>0x0000 0074</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>Settable</td>
<td>LPTIM2</td>
<td>LP timer 2 global interrupt</td>
<td>0x0000 0078</td>
</tr>
<tr>
<td>15</td>
<td>18</td>
<td>Settable</td>
<td>TIM1_BRK,</td>
<td>Timer 1 break, update, trigger and communication, capture compare interrupt</td>
<td>0x0000 007C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM1_UP,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM1_TRG_COM,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIM1_CC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>19</td>
<td>Settable</td>
<td>TIM2</td>
<td>Timer 2 global interrupt</td>
<td>0x0000 0080</td>
</tr>
<tr>
<td>17</td>
<td>20</td>
<td>Settable</td>
<td>TIM16</td>
<td>Timer 16 global interrupt</td>
<td>0x0000 0084</td>
</tr>
<tr>
<td>18</td>
<td>21</td>
<td>Settable</td>
<td>TIM17</td>
<td>Timer 17 global interrupt</td>
<td>0x0000 0088</td>
</tr>
<tr>
<td>19</td>
<td>22</td>
<td>Settable</td>
<td>IPCC_C2_RX_IT</td>
<td>IPCC CPU2 RX occupied interrupt</td>
<td>0x0000 008C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IPCC_C2_TX_IT</td>
<td>IPCC CPU2 TX free interrupt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HSEM</td>
<td>Semaphore interrupt 1 CPU2</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>23</td>
<td>Settable</td>
<td>PKA</td>
<td>Private key accelerator interrupt (C2IMR1[8])</td>
<td>0x0000 0090</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>True RNG</td>
<td>True random number generator interrupt (C2IMR1[9])</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AES1</td>
<td>AES1 global interrupt (C2IMR1[10])</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>24</td>
<td>Settable</td>
<td>AES2</td>
<td>AES2 global interrupt</td>
<td>0x0000 0094</td>
</tr>
<tr>
<td>22</td>
<td>25</td>
<td>Settable</td>
<td>LCD</td>
<td>LCD global interrupt (C2IMR2[22])</td>
<td>0x0000 0098</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>802_IT1</td>
<td>802.15.4 interrupt 1</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>26</td>
<td>Settable</td>
<td>I2C1_EVT</td>
<td>I2C1 event interrupt</td>
<td>0x0000 009C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I2C1_ER</td>
<td>I2C1 error interrupt</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>27</td>
<td>Settable</td>
<td>I2C3_EVT</td>
<td>I2C3 event interrupt</td>
<td>0x0000 00A0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I2C3_ERV</td>
<td>I2C3 error interrupt</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>28</td>
<td>Settable</td>
<td>SPI1</td>
<td>SPI1 global interrupt</td>
<td>0x0000 00A4</td>
</tr>
<tr>
<td>26</td>
<td>29</td>
<td>Settable</td>
<td>SPI2</td>
<td>SPI2 global interrupt</td>
<td>0x0000 00A8</td>
</tr>
<tr>
<td>27</td>
<td>30</td>
<td>Settable</td>
<td>USART1</td>
<td>USART1 global interrupt</td>
<td>0x0000 00AC</td>
</tr>
<tr>
<td>28</td>
<td>31</td>
<td>Settable</td>
<td>LPUART1</td>
<td>LPUART global interrupt</td>
<td>0x0000 00B0</td>
</tr>
<tr>
<td>29</td>
<td>32</td>
<td>Settable</td>
<td>SAI1</td>
<td>SAI1 A and B global interrupt</td>
<td>0x0000 00B4</td>
</tr>
<tr>
<td>30</td>
<td>33</td>
<td>Settable</td>
<td>BLE_BLUE_IT</td>
<td>BLE blue controller interrupt</td>
<td>0x0000 00B8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLE_RFC_IT</td>
<td>BLE radio control interrupt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLE_RFFMS_IT</td>
<td>BLE radio states interrupt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLE_HOST_WKUP</td>
<td>BLE host wakeup interrupt</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>34</td>
<td>Settable</td>
<td>802_IT2</td>
<td>802.15.4 interrupt 2</td>
<td>0x0000 00BC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>802_HOST_WKUP</td>
<td>802.15.4 host wakeup interrupt</td>
<td></td>
</tr>
</tbody>
</table>
13.5 Interrupt list

The device wakeup sources are listed in Table 62. Depending on its origin, the wakeup is handled according to different types, see Section 14.5: EXTI functional behavior for more information.

Some wakeup sources are able to generate an event to the CPUs. see Event column. The wakeup source capability to wakeup CPU1 and or CPU2 is listed in Wakeup column.

For CPUs interrupt handling see Section 13: Nested vectored interrupt controller (NVIC).

For Wakeup handling see Section 14: Extended interrupt and event controller (EXTI).

Table 62. Wakeup interrupt table

<table>
<thead>
<tr>
<th>EXTI no.</th>
<th>Acronym</th>
<th>Description</th>
<th>EXTI type</th>
<th>Event</th>
<th>Wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EXTI[0]</td>
<td>EXTI line 0 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>1</td>
<td>EXTI[1]</td>
<td>EXTI line 1 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>2</td>
<td>EXTI[2]</td>
<td>EXTI line 2 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>3</td>
<td>EXTI[3]</td>
<td>EXTI line 3 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>4</td>
<td>EXTI[4]</td>
<td>EXTI line 4 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>5</td>
<td>EXTI[5]</td>
<td>EXTI line 5 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>6</td>
<td>EXTI[6]</td>
<td>EXTI line 6 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>7</td>
<td>EXTI[7]</td>
<td>EXTI line 7 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>8</td>
<td>EXTI[8]</td>
<td>EXTI line 8 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>9</td>
<td>EXTI[9]</td>
<td>EXTI line 9 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>10</td>
<td>EXTI[10]</td>
<td>EXTI line 10 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>11</td>
<td>EXTI[11]</td>
<td>EXTI line 11 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>12</td>
<td>EXTI[12]</td>
<td>EXTI line 12 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>13</td>
<td>EXTI[13]</td>
<td>EXTI line 13 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>14</td>
<td>EXTI[14]</td>
<td>EXTI line 14 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>15</td>
<td>EXTI[15]</td>
<td>EXTI line 15 from SYSCFG</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>16</td>
<td>PVD</td>
<td>PVD line</td>
<td>Configurable A</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>17</td>
<td>RTC_ALARM</td>
<td>RTC Alarms (A and B) interrupt</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>18</td>
<td>TAMP_RTC_STAMP, LSE_CSS</td>
<td>RTC Tamper interrupt RTC TimeStamp interrupt RCC LSECSS interrupt</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>19</td>
<td>RTC_WKUP</td>
<td>RTC wakeup interrupt</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>20</td>
<td>COMP1</td>
<td>COMP1 line</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>21</td>
<td>COMP2</td>
<td>COMP2 line</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>22</td>
<td>I2C1 wakeup</td>
<td>I2C1 wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>23</td>
<td>I2C3 wakeup</td>
<td>I2C3 wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
</tbody>
</table>
### Table 62. Wakeup interrupt table (continued)

<table>
<thead>
<tr>
<th>EXTI no.</th>
<th>Acronym</th>
<th>Description</th>
<th>EXTI type</th>
<th>Event</th>
<th>Wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>USART1</td>
<td>USART1 wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>25</td>
<td>LPUART1</td>
<td>LPUART1 wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>USB wakeup</td>
<td>USB wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>29</td>
<td>LPTIM1</td>
<td>LP timer 1 wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>30</td>
<td>LPTIM2</td>
<td>LP timer 2 wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>31</td>
<td>PVM1</td>
<td>PVM1 line</td>
<td>Configurable A</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>32</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>33</td>
<td>PVM3</td>
<td>PVM3 line</td>
<td>Configurable A</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>34</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>36</td>
<td>IPCC CPU1 interrupts</td>
<td>IPCC CPU1 RX occupied and TX free interrupts</td>
<td>Direct C</td>
<td>No</td>
<td>CPU1(1)</td>
</tr>
<tr>
<td>37</td>
<td>IPCC CPU2 interrupts</td>
<td>IPCC CPU2 RX occupied and TX free interrupts</td>
<td>Direct C</td>
<td>No</td>
<td>CPU2(2)</td>
</tr>
<tr>
<td>38</td>
<td>HSEM interrupt 0</td>
<td>Semaphore interrupt 0 for CPU1</td>
<td>Direct C</td>
<td>No</td>
<td>CPU1(1)</td>
</tr>
<tr>
<td>39</td>
<td>HSEM interrupt 1</td>
<td>Semaphore interrupt 1 for CPU2</td>
<td>Direct C</td>
<td>No</td>
<td>CPU2(2)</td>
</tr>
<tr>
<td>40</td>
<td>C2SEV</td>
<td>CPU2 SEV line</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU2(3)</td>
</tr>
<tr>
<td>41</td>
<td>C1SEV</td>
<td>CPU1 SEV line</td>
<td>Configurable A</td>
<td>Yes</td>
<td>CPU2(4)</td>
</tr>
<tr>
<td>42</td>
<td>Flash interrupt</td>
<td>Flash ECC and global interrupts</td>
<td>Direct C</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>43</td>
<td>LCD wakeup</td>
<td>LCD wakeup</td>
<td>Direct B</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>44</td>
<td>HSE CSS interrupt</td>
<td>RCC HSE CSS interrupt</td>
<td>Direct C</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
<tr>
<td>45</td>
<td>BLE interrupts</td>
<td>BLE, RADIO, &amp; RF_FSM interrupts</td>
<td>Direct B</td>
<td>No</td>
<td>CPU2(2)</td>
</tr>
<tr>
<td>46</td>
<td>802 interrupts</td>
<td>802.15.4 Int0 and Int1 interrupts</td>
<td>Direct B</td>
<td>No</td>
<td>CPU2(2)</td>
</tr>
<tr>
<td>47</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>48</td>
<td>CDBGPWRUPREQ</td>
<td>Debug power up request wakeup</td>
<td>Direct</td>
<td>No</td>
<td>CPU1 and CPU2</td>
</tr>
</tbody>
</table>

1. For correct operation the EXTI direct event C2IMRx.IMn bit must be set to 0 before CPU1 uses this direct event.
2. For correct operation the EXTI direct event C1IMRx.IMn bit must be set to 0 before CPU2 uses this direct event.
3. For correct operation the EXTI configurable event both C2IMRx.IMn and C2EMRx.EMn bits must be set to 0 before CPU1 uses this configurable event.
4. For correct operation the EXTI configurable event both C1IMRx.IMn and C1EMRx.EMn bits must be set to 0 before CPU2 uses this configurable event.
14 Extended interrupt and event controller (EXTI)

The Extended interrupt and event controller (EXTI) manages the individual CPU and system wakeup through configurable and direct event inputs. It provides wakeup requests to the power control, and generates an interrupt request to the CPUs interrupt controller and events to the CPUs event input. For each CPU an additional event generation block (EVE) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from STOP modes, and the CPU to be woken up from the CSTOP and CSTANDBY modes.

The interrupt request and event request generation can also be used in RUN modes.

14.1 EXTI main features

The EXTI main features are the following:

- 49 input events supported
- Two CPUs supported
- All event inputs allow to wake up the system
- Events which do not have an associated wakeup flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI
- Some events can be used to generate a CPU wakeup event

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse)
  - Selectable active trigger edge
  - Interrupt pending status register bit.
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
  - SW trigger possibility

- Direct events (interrupt and wakeup sources from peripherals having an associated flag which requiring to be cleared in the peripheral)
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup and event generation
  - No SW trigger possibility

14.2 EXTI implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input events</td>
<td>0 to 48</td>
<td>0 to 42, 44 to 48</td>
</tr>
</tbody>
</table>
14.3 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input Trigger block, and the masking block as shown in Figure 30.

The register block contains all the EXTI registers.

The event input trigger block provides 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.

Figure 30. EXTI block diagram

Table 64. EXTI pin overview

<table>
<thead>
<tr>
<th>Pin 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 supports 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>it_exti_per(y)</td>
<td>O</td>
<td>Interrupts to the CPU1 to CPU(m) associated with Configurable event(y).</td>
</tr>
<tr>
<td>c(m)_evt_exti</td>
<td>O</td>
<td>High level sensitive event output for CPU(m) synchronous to hclk. (m= 1 to 2)</td>
</tr>
<tr>
<td>c(m)_evt_rst</td>
<td>I</td>
<td>Asynchronous reset input to clear c(m)_evt_exti. (m= 1 to 2)</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(m)_wakeup</td>
<td>O</td>
<td>Wakeup request to PWR for CPU(m), synchronous to hclk. (m= 1 to 2)</td>
</tr>
</tbody>
</table>
14.3.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in STOP mode or a CPU is in CSTOP mode are connected to the EXTI.

Peripheral wakeup signals that generate a pulse or which do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events the EXTI provides a status pending bit that has 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 that has to be cleared in the peripheral, are connected to an EXTI direct event input. 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.

The EXTI configurable event interrupts are connected to the respective interrupt controller of each CPU(m).

The dedicated EXTI/EVG CPU(m) event is connected to the respective CPU(m) rxev input.

The EXTI CPU(m) wakeup signals are connected to the PWR block, and are used to wake up the system and CPU(m) sub-system bus clocks.

<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 Events 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>
14.4 EXTI functional description

Depending on the EXTI event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by rising edge selection
  EXTI rising trigger selection register (EXTI_RTSR1),
  EXTI rising trigger selection register (EXTI_RTSR2),
  and falling edge selection
  EXTI falling trigger selection register (EXTI_FTSR1),
  EXTI falling trigger selection register (EXTI_FTSR2).

- Software trigger, by
  EXTI software interrupt event register (EXTI_SWIER1),
  EXTI software interrupt event register (EXTI_SWIER2).

- Interrupt pending flag, by
  EXTI pending register (EXTI_PR1),
  EXTI pending register (EXTI_PR2).

- CPU wakeup and interrupt enable, by
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR1),
  EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1),
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR2),
  EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2).

- CPU wakeup and event enable, by
  EXTI CPU wakeup with event mask register (EXTI_EMR1),
  EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1)
  EXTI CPU wakeup with event mask register (EXTI_EMR2),
  EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2).

Table 66. EXTI event input configurations and register control

<table>
<thead>
<tr>
<th>Event input type</th>
<th>Logic implementation</th>
<th>EXTI_RTSR</th>
<th>EXTI_FTSR</th>
<th>EXTI_SWIER</th>
<th>EXTI_PR</th>
<th>EXTI_C2IMR1</th>
<th>EXTI_C2EMR1</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>

1. Only for input events with configuration "rxev generation" enabled.

14.4.1 EXTI configurable event input wakeup

Figure 31 is a detailed representation of the logic associated with configurable event inputs which will wake up the CPU(m) sub-system bus clocks and generated an EXTI pending flag and interrupt to the CPU(m) and or a CPU(m) wakeup event.
1. Only for the input events that support CPU rxev generation c(n)_event.

The software interrupt event register allows to trigger 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.

Each CPU has its dedicated interrupt mask register and a dedicated event mask registers. The enabled event allows to generate an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The event pending register (EXTI_PR) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPUs. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to all CPUs. The configurable event interrupts need to be acknowledged by software in the EXTI_PR register.

When a CPU(m) interrupt or CPU(m) event is enabled the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

**Note:** A detected configurable event interrupt pending request, may be cleared by any CPU. The system is unable to enter into Low-power modes as long as an interrupt pending request is active.
14.4.2 EXTI direct event input wakeup

*Figure 32* 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 associated Peripheral interrupt flag is set.

**Note:** The direct events are cleared in the peripheral generating the event. When a direct event input enabled by CPU(m) is cleared by another CPU before the CPU(m) clock is running, the CPU(m) no longer receives a CPU(m) interrupt nor a CPU(m) event and does not wake up. However, the system stays in RUN mode, generating the CPU(m) clock. For this reason CPU(m) direct events must NOT be cleared by the other CPU.

---

1. Only for the input events that support CPU rxev generation c(n)_event.

14.5 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(m) wakeup is conditioned by the CPU(m) interrupt mask and CPU(m) event mask.
For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU(m) interrupt is unmasked the corresponding pending bit EXTI_PR.PIFn is set and the CPU(m) sub-system is woken up and CPU interrupt signal is activated. The EXTI_PR.PIFn pending bit must be cleared by software writing it to ‘1’. This clears the CPU interrupt.

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(m) interrupt is unmasked the corresponding CPU sub-system is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU(m) event must be unmasked to generate an event. When the enabled edge(s) occur on the event input a CPU(m) event pulse is generated. There is no event pending bit.

For the configurable event inputs an event request can be generated by software when writing a ‘1’ in the software interrupt/event register EXTI_SWIER, generating a rising edge on the event. The edge event pending bit is set in EXTI_PR, irrespective of the setting in EXTI_RTSR.

<table>
<thead>
<tr>
<th>CPU interrupt enable EXTI_CmIMR.IMn</th>
<th>CPU event enable EXTI_CmEMMR.EMn</th>
<th>Configurable event inputs EXTI_PR.PIFn</th>
<th>exti(n) interrupt(1)</th>
<th>CPU(m) event</th>
<th>CPU(m) wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 for all CPUs</td>
<td>0</td>
<td>No</td>
<td>Masked</td>
<td>Masked</td>
<td>Masked</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Status latched</td>
<td>Yes</td>
<td>Masked</td>
<td>Yes</td>
</tr>
<tr>
<td>1 for any CPU</td>
<td>0</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. The single exti(n) interrupt goes to all CPUs. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU(m) interrupt controller.
2. Only if CPU(m) interrupt is enabled in EXTI_CmIMR.IMn.
14.6 EXTI registers

The EXTI register map is divided as detailed in Table 68.

Table 68. EXTI register map 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>
<tr>
<td>0x020 - 0x03C</td>
<td>General configurable event [63:32] configuration</td>
</tr>
<tr>
<td>0x040 - 0x05C</td>
<td>General configurable event [95:64] configuration</td>
</tr>
<tr>
<td>0x080 - 0x0BC</td>
<td>CPU1 input event configuration</td>
</tr>
<tr>
<td>0x0C0 - 0x0FC</td>
<td>CPU2 input event configuration</td>
</tr>
</tbody>
</table>

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

14.6.1 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>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</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></td>
<td></td>
<td></td>
<td></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 14 13 12 11 10  9  8  7  6  5  4  3  2  1  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>
<td></td>
</tr>
<tr>
<td>RT15</td>
<td>RT14</td>
<td>RT13</td>
<td>RT12</td>
<td>RT11</td>
<td>RT10</td>
<td>RT9</td>
<td>RT8</td>
<td>RT7</td>
<td>RT6</td>
<td>RT5</td>
<td>RT4</td>
<td>RT3</td>
<td>RT2</td>
<td>RT1</td>
<td>RT0</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 31 **RT31**: Rising trigger event configuration bit of configurable Event input 31\(^{(1)}\).
0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:0 **RT[21:0]**: Rising trigger event configuration bit of configurable event input x (x = 21 to 0)\(^{(1)}\).
0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.
14.6.2 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>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>

Bit 31  **FT31**: Falling trigger event configuration bit of configurable event input 31\(^{(1)}\).

0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:0  **FT[21:0]**: Falling trigger event configuration bit of configurable event input x (x = 21 to 0)\(^{(1)}\).

0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

14.6.3 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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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  **SWI31**: Software interrupt on event 31

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger a rising edge event on event 31.
This bit is auto cleared by HW.

Bits 30:22 Reserved, must be kept at reset value.
14.6.4 EXTI pending register (EXTI_PR1)

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>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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 PIF31: configurable event inputs 31 Pending bit.
0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:0 PIF[21:0]: configurable event inputs x (x = 21 to 0) Pending bit.
0: No trigger request occurred
1: Selected trigger request occurred
This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

14.6.5 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x020
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>RT33</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>
<td></td>
</tr>
</tbody>
</table>

Bits 31:10 Reserved, must be kept at reset value.
14.6.6 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x024
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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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.

Bit 9 FT41: Falling trigger event configuration bit of configurable event input 41\(^{(1)}\)
- 0: Falling trigger disabled (for event and interrupt) for input line
- 1: Falling trigger enabled (for event and interrupt) for input line

Bit 8 FT40: Falling trigger event configuration bit of configurable event input 40\(^{(1)}\)
- 0: Falling trigger disabled (for event and interrupt) for input line
- 1: Falling trigger enabled (for event and interrupt) for input line

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

Bit 1 FT33: Falling trigger event configuration bit of configurable event input 33\(^{(1)}\)
- 0: Falling trigger disabled (for event and interrupt) for input line
- 1: Falling trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable Event input. In this case, both edges generate a trigger.
14.6.7 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x028
Reset value: 0x0000 0000
Contains only register bits for configurable events.

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

Bit 9 **SWI41**: Software interrupt on event 41
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger a rising edge event on event 41.
This bit is auto cleared by HW.

Bit 8 **SWI40**: Software interrupt on event 40
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger a rising edge event on event 40.
This bit is auto cleared by HW.

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

Bit 1 **SWI33**: Software interrupt on event 33
A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit will trigger a rising edge event on event 33.
This bit is auto cleared by HW.

Bit 0 Reserved, must be kept at reset value.
14.6.8 EXTI pending register (EXTI_PR2)

Address offset: 0x02C
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></td>
<td></td>
<td></td>
<td></td>
<td></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>rc_w1</td>
<td>rc_w1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 9 PIF41: configurable event inputs 41 pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 8 PIF40: configurable event inputs 40 pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

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

Bit 1 PIF33: configurable event inputs 33 pending bit
0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 0 Reserved, must be kept at reset value.

14.6.9 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

Address offset: 0x080
Reset value: 0x7FC0 0000

Contains register bits for configurable events and direct 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></td>
<td></td>
<td></td>
<td></td>
<td></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>IM31</td>
<td>IM30</td>
<td>IM29</td>
<td>IM28</td>
<td>IM27</td>
<td>IM26</td>
<td>IM25</td>
<td>IM24</td>
<td>IM23</td>
<td>IM22</td>
<td>IM21</td>
<td>IM20</td>
<td>IM19</td>
<td>IM18</td>
<td>IM17</td>
<td>IM16</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>
<tr>
<td>IM15</td>
<td>IM14</td>
<td>IM13</td>
<td>IM12</td>
<td>IM11</td>
<td>IM10</td>
<td>IM9</td>
<td>IM8</td>
<td>IM7</td>
<td>IM6</td>
<td>IM5</td>
<td>IM4</td>
<td>IM3</td>
<td>IM2</td>
<td>IM1</td>
<td>IM0</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 31:0 **IM[31:0]**: CPU wakeup with interrupt mask on event input x (x = 31:0)
1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

### 14.6.10 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1)

Address offset: 0x0C0  
Reset value: 0x7FC0 0000

Contains register bits for configurable events and direct 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>IM31</td>
<td>IM30</td>
<td>IM29</td>
<td>IM28</td>
<td>IM27</td>
<td>IM26</td>
<td>IM25</td>
<td>IM24</td>
<td>IM23</td>
<td>IM22</td>
<td>IM21</td>
<td>IM20</td>
<td>IM19</td>
<td>IM18</td>
<td>IM17</td>
<td>IM16</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>IM15</td>
<td>IM14</td>
<td>IM13</td>
<td>IM12</td>
<td>IM11</td>
<td>IM10</td>
<td>IM9</td>
<td>IM8</td>
<td>IM7</td>
<td>IM6</td>
<td>IM5</td>
<td>IM4</td>
<td>IM3</td>
<td>IM2</td>
<td>IM1</td>
<td>IM0</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 31:0 **IM[31 :0]**: CPU2 wakeup with interrupt mask on event input x (x = 31 to 0)
1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

### 14.6.11 EXTI CPU wakeup with event mask register (EXTI_EMR1)

Address offset: 0x084

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>Bits 31:22 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>
<td></td>
</tr>
<tr>
<td>Bits 21:17 <strong>EM[21:17]</strong>: CPU wakeup with event generation mask on event input x (x = 21 to 17)</td>
<td></td>
<td></td>
<td></td>
<td></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>0: Wakeup with event generation from Line x is masked</td>
<td></td>
<td></td>
<td></td>
<td></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>1: Wakeup with event generation from Line x is unmasked</td>
<td></td>
<td></td>
<td></td>
<td></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 16 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>
<td></td>
</tr>
<tr>
<td>Bits 15:0 <strong>EM[15:0]</strong>: CPU wakeup with event generation mask on event input x (x = 15 to 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>
<td></td>
</tr>
<tr>
<td>0: Wakeup with event generation from Line x is masked</td>
<td></td>
<td></td>
<td></td>
<td></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>1: Wakeup with event generation from Line x is unmasked</td>
<td></td>
<td></td>
<td></td>
<td></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>
14.6.12 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1)

Address offset: 0x0C4
Reset value: 0x0000 0000

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

Bits 21:17 **EM[21:17]:** CPU2 wakeup with event generation mask on event input x (x = 21 to 17)
- 0: Wakeup with event generation from Line x is masked
- 1: Wakeup with event generation from Line x is unmasked

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **EM[15:0]:** CPU2 wakeup with event generation mask on event input x (x = 15 to 0)
- 0: Wakeup with event generation from Line x is masked
- 1: Wakeup with event generation from Line x is unmasked

14.6.13 EXTI CPU wakeup with interrupt mask register (EXTI_IMR2)

Address offset: 0x090
Reset value: 0x0001 FCFD

Contains register bits for configurable events and direct events.

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

Bits 16:0 **IM[48:32]:** CPU wakeup with interrupt mask on event input x (x = 48 to 32)\(^{(1)}\)(\(^{(2)}\))
- 0: Wakeup with interrupt request from Line x is masked
- 1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.
14.6.14 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2)

Address offset: 0x0D0
Reset value: 0x0001 FCFD

Contains register bits for configurable events and direct 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>IM47</td>
<td>IM46</td>
<td>IM45</td>
<td>IM44</td>
<td>IM43</td>
<td>IM42</td>
<td>IM41</td>
<td>IM40</td>
<td>IM39</td>
<td>IM38</td>
<td>IM37</td>
<td>IM36</td>
<td>IM35</td>
<td>IM34</td>
<td>IM33</td>
<td>IM32</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 31:17 Reserved, must be kept at reset value.

Bits 16:0 IM[48:32]: CPU wakeup with interrupt mask on event input x (x = 48 to 32)(1)(2)
   0: Wakeup with interrupt request from Line x is masked
   1: Wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

14.6.15 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>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></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>
</tbody>
</table>

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

Bit 9 EM41: CPU wakeup with event generation mask on event input 41
   0: Wakeup with event generation from Line 41 is masked
   1: Wakeup with event generation from Line 41 is unmasked

Bit 8 EM40: CPU wakeup with event generation mask on event input 40.
   0: Wakeup with event generation from Line 40 is masked
   1: Wakeup with event generation from Line 40 is unmasked

Bits 7:0 Reserved, must be kept at reset value.
### 14.6.16 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2)

Address offset: 0x0D4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:10</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 9</td>
<td><strong>EM41</strong>: CPU2 wakeup with event generation mask on event input 41</td>
</tr>
<tr>
<td></td>
<td>0: Wakeup with event generation from Line 41 is masked</td>
</tr>
<tr>
<td></td>
<td>1: Wakeup with event generation from Line 41 is unmasked</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>EM40</strong>: CPU2 wakeup with event generation mask on event input 40.</td>
</tr>
<tr>
<td></td>
<td>0: Wakeup with event generation from Line 40 is masked</td>
</tr>
<tr>
<td></td>
<td>1: Wakeup with event generation from Line 40 is unmasked</td>
</tr>
<tr>
<td>Bits 7:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
## 14.6.17 EXTI register map

The following table gives the EXTI register map and the reset values.

### Table 69. EXTI 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  | EXTI_RTSR1  | RT[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x004  | EXTI_FTSR1  | FT[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x008  | EXTI_SWIER1 | SWI[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x00C  | EXTI_PR1    | PIF[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x010-0x01C | Reserved |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x020  | EXTI_RTSR2  | RT[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x024  | EXTI_FTSR2  | FT[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x028  | EXTI_SWIER2 | SWI[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x02C  | EXTI_PR2    | PIF[1:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x030-0x07C | Reserved |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x080  | EXTI_IMR1   | IM[31:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x084  | EXTI_EMR1   | EM[21:17] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x088-0x08C | Reserved |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x090  | EXTI_IMR2   | IM[48:32] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 1     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x094  | EXTI_EMR2   | EM[4:0] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value | 0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x098-0x0BC | Reserved |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
Table 69. EXTI 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  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0C0  | EXTI_C2IMR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C4  | EXTI_C2EMR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EM[21:17]| EM[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  | 0  |
| 0x0C8- | EXTI_C2MR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0CC  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0D0  | EXTI_C2MR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0D4  | EXTI_C2EMR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | EM[43:40]|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.2 on page 66 for the register boundary addresses.
15 Quad-SPI interface (QUADSPI)

15.1 Introduction

The QUADSPI is a specialized communication interface targeting single, dual- or quad-SPI flash memories. It can operate in any of the three following modes:

- **indirect mode**: all the operations are performed using the QUADSPI registers.
- **automatic status-polling mode**: the external flash memory status register is periodically read and an interrupt can be generated in case of flag setting.
- **memory-mapped mode**: the external flash memory is mapped to the device address space and is seen by the system as if it was an internal memory.

15.2 QUADSPI main features

- Three functional modes: indirect, automatic status-polling, and memory-mapped
- SDR and DDR support
- Fully programmable opcode for both indirect and memory-mapped modes
- Fully programmable frame format for both indirect and memory-mapped modes
- Integrated FIFO for reception and transmission
- 8-, 16-, and 32-bit data accesses allowed
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error

15.3 QUADSPI functional description

15.3.1 QUADSPI block diagram

*Figure 33. QUADSPI block diagram*
15.3.2 QUADSPI pins

The table below lists the QUADSPI pins for interfacing with a flash memory.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>Digital output</td>
<td>Clock to flash memory</td>
</tr>
<tr>
<td>BK1_IO0/SO</td>
<td>Digital input/output</td>
<td>Bidirectional I/O in dual/quad modes or serial output in single mode</td>
</tr>
<tr>
<td>BK1_IO1/SI</td>
<td>Digital input/output</td>
<td>Bidirectional I/O in dual/quad modes or serial input in single mode</td>
</tr>
<tr>
<td>BK1_IO2</td>
<td>Digital input/output</td>
<td>Bidirectional I/O in quad mode</td>
</tr>
<tr>
<td>BK1_IO3</td>
<td>Digital input/output</td>
<td>Bidirectional I/O in quad mode</td>
</tr>
<tr>
<td>BK1_NCS</td>
<td>Digital output</td>
<td>Chip select (active low)</td>
</tr>
</tbody>
</table>

15.3.3 QUADSPI command sequence

The QUADSPI communicates with the flash memory using commands. Each command can include five phases: instruction, address, alternate byte, dummy, data. Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phase must be present.

NCS falls before the start of each command and rises again after each command finishes.

Figure 34. Example of read command in quad-SPI mode

Instruction phase

During this phase, an 8-bit instruction, configured in INSTRUCTION bitfield of QUADSPI_CCR[7:0] register, is sent to the flash memory, specifying the type of operation to be performed.

Most flash memories can receive instructions only one bit at a time from the IO0/SO signal (single-SPI mode), the instruction phase can optionally send 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the IMODE[1:0] bitfield of QUADSPI_CCR[9:8] register.

When IMODE = 00, the instruction phase is skipped, and the command sequence starts with the address phase, if present.
Address phase

In the address phase, 1-4 bytes are sent to the flash memory to indicate the address of the operation. The number of address bytes to be sent is configured in the ADSIZE[1:0] bitfield of QUADSPI_CCR[13:12] register. In indirect and automatic status-polling modes, address bytes to be sent are specified in the ADDRESS[31:0] bitfield of QUADSPI_AR register, while in memory-mapped mode, the address is given directly via the AHB (from the Cortex or from a DMA).

The address phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ADMODE[1:0] bitfield of QUADSPI_CCR[11:10] register.

When ADMODE = 00, the address phase is skipped, and the command sequence proceeds directly to the next phase, if any.

Alternate-byte phase

In the alternate-byte phase, 1-4 bytes are sent to the flash memory, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the [1:0] bitfield of QUADSPI_CCR[17:16] register. The bytes to be sent are specified in the QUADSPI_ABR register.

The alternate-bytes phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] bitfield of QUADSPI_CCR[15:14] register.

When ABMODE = 00, the alternate-byte phase is skipped, and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-mode is used and only two cycles are used for the alternate byte. In this case, the firmware can use quad-mode (ABMODE = 11) and send a byte with bits 7 and 3 of ALTERNATE set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low). In this case, the upper two bits of the nibble to be sent are placed in bits 4:3 of ALTERNATE, while the lower two bits are placed in bits 1 and 0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE must be set to 0x8A (1000_1010).

Dummy-cycle phase

In the dummy-cycle phase, 1-31 cycles are given without any data being sent or received, in order to give time to the flash memory to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] bitfield of QUADSPI_CCR[22:18] register. In both SDR and DDR modes, the duration is specified as a number of full CLK cycles.

When DCYC is zero, the dummy-cycles phase is skipped, and the command sequence proceeds directly to the data phase, if present.

The operating mode of the dummy-cycles phase is determined by DMODE.

In order to assure enough “turn-around” time for changing data signals from output mode to input mode, there must be at least one dummy cycle when using dual or quad mode to receive data from the flash memory.
Data phase

During the data phase, any number of bytes can be sent to, or received from the flash memory.

In indirect and automatic status-polling modes, the number of bytes to be sent/received is specified in the QUADSPI_DLR register.

In indirect-write mode the data to be sent to the flash memory must be written to the QUADSPI_DR register. In indirect-read mode the data received from the flash memory is obtained by reading the QUADSPI_DR register.

In memory-mapped mode, the data which is read is sent back directly over the AHB to the Cortex or to a DMA.

The data phase can send/receive 1 bit at a time (over SO/SI in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] bitfield of QUADSPI_CCR[15:14] register.

When DMODE = 00, the data phase is skipped, and the command sequence finishes immediately by raising NCS. This configuration must only be used in only indirect write mode.

15.3.4 QUADSPI signal interface protocol modes

Single-SPI mode

This legacy SPI mode allows just one single bit to be sent/received serially. In this mode, data are sent to the flash memory over the SO signal (whose I/O shared with IO0). Data received from the flash memory arrive via SI (whose I/O shared with IO1).

The different phases can each be configured separately to use this mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in QUADSPI_CCR) to 01.

In each phase which is configured in single-SPI mode:
- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).

This is the case even for the dummy phase if DMODE = 01.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 10.

In each phase which is configured in dual-SPI mode:
- IO0/IO1 are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.

In the dummy phase when DMODE = 01, IO0/IO1 are always high-impedance.
Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use quad-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 11.

In each phase which is configured in this mode, IO0/IO1/IO2/IO3 are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE = 11, IO0/IO1/IO2/IO3 are all high-impedance.

IO2 and IO3 are used only in quad-SPI mode. If none of the phases are configured to use quad-SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while the QUADSPI is active.

SDR mode

By default, the DDRM bit (QUADSPI_CCR[31]) is 0 and the QUADSPI operates in single-data rate (SDR) mode.

In SDR mode, when the QUADSPI drives IO0/SO, IO1, IO2, IO3 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the QUADSPI assumes that flash memories also send the data using CLK falling edge. By default (when SSHIFT = 0), the signals are sampled using the following (rising) edge of CLK.

DDR mode

When the DDRM bit (QUADSPI_CCR[31]) is set to 1, the QUADSPI operates in double-data rate (DDR) mode.

In DDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals in the address/alternate-byte/data phases, a bit is sent on each of CLK falling and rising edges.

The instruction phase is not affected by DDRM. The instruction is always sent using CLK falling edge.

When receiving data in DDR mode, the QUADSPI assumes that flash memories also send the data using both CLK rising and falling edges. When DDRM = 1, the firmware must clear SSHIFT (bit 4 of QUADSPI_CR). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).
15.3.5 QUADSPI indirect mode

When in indirect mode, commands are started by writing to QUADSPI registers, and data are transferred by writing or reading the data register, in the same way as for other communication peripherals.

When FMODE = 00 (QUADSPI_CCR[27:26]), the QUADSPI is in indirect-write mode, where bytes are sent to the flash memory during the data phase. Data are provided by writing to the data register (QUADSPI_DR).

When FMODE = 01, the QUADSPI is in indirect-read mode, where bytes are received from the flash memory during the data phase. Data are recovered by reading QUADSPI_DR.

The number of bytes to be read/written is specified in the data length register (QUADSPI_DLR). If QUADSPI_DLR = 0xFFFF_FFFF (all 1s), then the data length is considered undefined and the QUADSPI simply continues to transfer data until the end of flash memory (as defined by FSIZE) is reached. If no bytes are to be transferred, DMODE (QUADSPI_CCR[25:24]) must be set to 00.

If QUADSPI_DLR = 0xFFFF_FFFF and FSIZE = 0x1F (max value indicating a 4-Gbyte flash memory), then in this special case, transfers continue indefinitely, stopping only after an abort request or after the QUADSPI is disabled. After the last memory address is read (at address 0xFFFF_FFFF), reading continues with address = 0x0000_0000.

When the programmed number of bytes to be transmitted or received is reached, TCF is set and an interrupt is generated if TCIE = 1. In the case of undefined number of data, the TCF is set when the limit of the external SPI memory is reached according to the flash memory size defined in QUADSPI_CR.
Triggering the start of a command

Essentially, a command starts as soon as firmware gives the last information that is necessary for this command. Depending on the QUADSPI configuration, there are three different ways to trigger the start of a command in indirect mode. The commands starts immediately:

- after a write is performed to INSTRUCTION[7:0] (QUADSPI_CCR), if no address is necessary (when ADMODE = 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to ADDRESS[31:0] (QUADSPI_AR), if an address is necessary (when ADMODE ≠ 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to DATA[31:0] (QUADSPI_DR), if data needs to be provided by the firmware (when FMODE = 00 and DMODE ≠ 00)

Writes to the alternate byte register (QUADSPI_ABR) never trigger the communication start. If alternate bytes are required, they must be programmed before.

As soon as a command is started, BUSY (bit 5 of QUADSPI_SR) is automatically set.

FIFO and data management

In indirect mode, data go through a 32-byte FIFO which is internal to the QUADSPI. FLEVEL[5:0] (QUADSPI_SR[13:8]) indicates how many bytes are currently being held in the FIFO.

In indirect-write mode (FMODE = 00), the firmware adds data to the FIFO when it writes QUADSPI_DR. Word writes add 4 bytes to the FIFO, halfword writes add 2 bytes, and byte writes add only 1 byte. If the firmware adds too many bytes to the FIFO (more than is indicated by DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

Byte/halfword accesses to QUADSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES[4:0] is used to define a FIFO threshold. When the threshold is reached, FTF (FIFO threshold flag) is set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there are data in the FIFO after the last byte is read from the flash memory, regardless of the FTHRES setting. In indirect-write mode, FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when FTF is set. If DMAEN = 1, a DMA transfer is initiated when FTF is set. FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data is transferred by the CPU or DMA).

15.3.6 QUADSPI automatic status-polling mode

In automatic status-polling mode, the QUADSPI periodically starts a command to read a defined number of status bytes (up to 4). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

Accesses to the flash memory begin in the same way as in indirect-read mode: if no address is required (AMODE = 00), accesses begin as soon as the QUADSPI_CCR is written.
Otherwise, if an address is required, the first access begins when QUADSPI_AR is written. BUSY goes high at this point and stays high even between the periodic accesses.

The contents of MASK[31:0] (QUADSPI_PSMAR) are used to mask the data from the flash memory in automatic status-polling mode. If MASK[n] = 0, then bit n of the result is masked and not considered. If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] (QUADSPI_PSMAR), then there is a match for bit n.

If the polling match mode bit (PMM, bit 23 of QUADSPI_CR) is 0, then “AND” match mode is activated. This means status match flag (SMF) is set only when there is a match on all of the unmasked bits.

If PMM = 1, then “OR” match mode is activated. This means SMF is set if there is a match on any of the unmasked bits.

An interrupt is called when SMF is set if SMIE = 1.

If the automatic status-polling mode stop (APMS) bit is set, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1, and the periodic accesses continue until there is an abort or the QUADSPI is disabled (EN = 0).

The data register (QUADSPI_DR) contains the latest received status bytes (the FIFO is deactivated). The content of the data register is not affected by the masking used in the matching logic. The FTF status bit is set as soon as a new reading of the status is complete, and FTF is cleared as soon as the data is read.

15.3.7 QUADSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

It is forbidden to access the quad-SPI flash bank area before having properly configured and enabled the QUADSPI peripheral.

No more than 256 Mbytes can addressed even if the flash memory capacity is larger.

If an access is made to an address outside of the range defined by FSIZE but still within the 256-Mbyte range, then a bus error is given. The effect of this error depends on the bus master that attempted the access:

- If it is the Cortex CPU, bus fault exception is generated when enabled (or an HardFault exception when bus fault is disabled).
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, halfword, and word access types are all supported.

Support for execute in place (XIP) operation is implemented, where the QUADSPI anticipates the next access and load in advance the byte at the following address. If the subsequent access is indeed made at a continuous address, the access is completed faster since the value is already prefetched.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without any access since when the FIFO becomes full with prefetch data.
BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is a timeout, there is an abort, or the peripheral is disabled.

15.3.8 QUADSPI flash memory configuration

The device configuration register (QUADSPI_DCR) can be used to specify the characteristics of the external SPI flash memory.

The FSIZE[4:0] bitfield defines the size of external memory using the following formula:

$$\text{Number of bytes in flash memory} = 2^{\text{FSIZE+1}}$$

FSIZE+1 is effectively the number of address bits required to address the flash memory. The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

When the QUADSPI executes two commands, one immediately after the other, it raises NCS high between the two commands for only one CLK cycle by default. If the flash memory requires more time between commands, the CSHT bitfield can be used to specify the minimum number of CLK cycles (up to 8) that NCS must remain high.

The clock mode (CKMODE) bit indicates the CLK signal logic level in between commands (when NCS = 1).

15.3.9 QUADSPI delayed data sampling

By default, the QUADSPI samples the data driven by the flash memory one half of a CLK cycle after the flash memory drives the signal.

In case of external signal delays, it may be beneficial to sample the data later. Using SSHIFT (bit 4 of QUADSPI_CR), the sampling of the data can be shifted by half of a CLK cycle.

Clock shifting is not supported in DDR mode: SSHIFT must be clear when DDRM bit is set.

15.3.10 QUADSPI configuration

The QUADSPI configuration is done in two phases:
1. QUADSPI peripheral configuration
2. QUADSPI flash memory configuration

Once configured and enabled, the QUADSPI can be used in one of its three operating modes: indirect, automatic status-polling, or memory-mapped mode.

QUADSPI configuration

The QUADSPI is configured using QUADSPI_CR. The user must configure the clock prescaler division factor and the sample shifting settings for the incoming data.

The DDR mode can be set through the DDRM bit. When setting the quad-SPI interface in DDR mode, the internal divider of kernel clock must be set with a division ratio of two or more. Once enabled, the address and the alternate bytes are sent on both clock edges, and the data are sent/received on both clock edges. Regardless of the DDRM bit setting, instructions are always sent in SDR mode.

DMA requests are enabled setting the DMAEN bit. In case of interrupt usage, their respective enable bit can be also set during this phase.
The FIFO level for either DMA request generation or interrupt generation is programmed in the FTHRES bits.

If a timeout counter is needed, the TCEN bit can be set and the timeout value programmed in the QUADSPI_LPTR register.

**QUADSPI flash memory configuration**

The parameters related to the targeted external flash memory are configured through the QUADSPI_DCR register. The user must program the flash memory size in the FSIZE bits, the chip-select minimum high time in CSHT bits, and the functional mode (Mode 0 or Mode 3) in the MODE bit.

### 15.3.11 QUADSPI use

The operating mode is selected using FMODE[1:0] (QUADSPI_CCR[27:26]).

**Indirect mode**

When FMODE is programmed to 00, the indirect-write mode is selected and data can be sent to the flash memory. With FMODE = 01, the indirect-read mode is selected where data can be read from the flash memory.

When the QUADSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in QUADSPI_DLR.
2. Specify the frame format, mode and instruction code in QUADSPI_CCR.
3. Specify optional alternate byte to be sent right after the address phase in QUADSPI_ABR.
4. Specify the operating mode in QUADSPI_CR. If FMODE = 00 (indirect-write mode) and DMAEN = 1, then QUADSPI_AR must be specified before QUADSPI_CR. Otherwise QUADSPI_DR can be written by the DMA before QUADSPI_AR is updated (if the DMA controller has already been enabled).
5. Specify the targeted address in QUADSPI_AR.
6. Read/write the data from/to the FIFO through QUADSPI_DR.

When writing QUADSPI_CR, the user specifies the following settings:
- enable bit (EN) set to 1
- DMA enable bit (DMAEN) for transferring data to/from RAM
- timeout counter enable bit (TCEN)
- sample shift setting (SSHIFT)
- FIFO threshold level (FTRHES) to indicate when the FTF flag must be set
- interrupt enables
- automatic status-polling mode parameters: match mode and stop mode (valid when FMODE = 11)
- clock prescaler

When writing QUADSPI_CCR, the user specifies the following parameters:
- instruction byte through INSTRUCTION bits
- the way the instruction has to be sent through the IMODE bits (1/2/4 lines)
- the way the address has to be sent through the ADMODE bits (None/1/2/4 lines)
Quad-SPI interface (QUADSPI)

- address size (8/16/24/32-bit) through ADSIZE bits
- the way the alternate bytes have to be sent through the ABMODE (None/1/2/4 lines)
- alternate bytes number (1/2/3/4) through the ABSIZE bits
- presence or not of dummy bytes through the DBMODE bit
- number of dummy bytes through the DCYC bits
- the way data have to be sent/received (none/1/2/4 lines) through DMODE bits

If neither QUADSPI_AR nor QUADSPI_DR need to be updated for a particular command, then the command sequence starts as soon as QUADSPI_CCR is written. This is the case when both ADMODE and DMODE are 00, or if just ADMODE = 00 when in indirect read mode (FMODE = 01).

When an address is required (ADMODE is not 00) and the data register does not need to be written (when FMODE = 01 or DMODE = 00), the command sequence starts as soon as the address is updated with a write to QUADSPI_AR.

In case of data transmission (FMODE = 00 and DMODE! = 00), the communication start is triggered by a write in the FIFO through QUADSPI_DR.

**Automatic status-polling mode**

This mode is enabled setting the FMODE bitfield (QUADSPI_CCR[27:26]) to 10. In this mode, the programmed frame is sent and the data retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in QUADSPI_DLR, it is ignored and only 4 bytes are read.

The periodicity is specified in the QUADSPI_PISR register.

Once the status data is retrieved, it can internally be processed:
- to set the status match flag and generate an interrupt if enabled
- to stop automatically the periodic retrieving of the status bytes

The received value can be masked with the value stored in QUADSPI_PSMKR and ORed or ANDed with the value stored in QUADSPI_PSMAR.

In case of match, the status match flag is set and an interrupt is generated if enabled, and the QUADSPI can be automatically stopped if the AMPS bit is set.

In any case, the latest retrieved value is available in QUADSPI_DR.

**Memory-mapped mode**

In memory-mapped mode, the external flash memory is seen as an internal memory but with some latency during accesses. Only read operations are allowed to the external flash memory in this mode.

The memory-mapped mode is entered by setting FMODE to 11 in QUADSPI_CCR.

The programmed instruction and frame is sent when a master is accessing the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate linear reads. Any access to QUADSPI_DR in this mode returns zero.

QUADSPI_DLR has no meaning in memory-mapped mode.
15.3.12 Sending the instruction only once

Some flash memories (for example: Winbound) provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. One can take advantage of such a feature using the SIOO bit (QUADSPI_CCR[28]).

SIOO is valid for all functional modes (indirect, automatic status-polling, and memory-mapped). If the SIOO bit is set, the instruction is sent only for the first command following a write to QUADSPI_CCR. Subsequent command sequences skip the instruction phase, until there is a write to QUADSPI_CCR.

SIOO has no effect when IMODE = 00 (no instruction).

15.3.13 QUADSPI error management

An error can be generated in the following case:

- In indirect mode or automatic status-polling mode when a wrong address is programmed in QUADSPI_AR (according to the flash memory size defined by FSIZE[4:0] in the QUADSPI_DCR), TEF is set and an interrupt is generated if enabled.
- Also in indirect mode, if the address plus the data length exceeds the flash memory size, TEF is set as soon as the access is triggered.
- In memory-mapped mode, when an out-of-range access is done by a master or when the QUADSPI is disabled, a bus error is generated as a response to the faulty bus master request.
- When a master is accessing the memory mapped space while the memory-mapped mode is disabled, a bus error is generated as a response to the faulty bus master request.

15.3.14 QUADSPI busy bit and abort functionality

Once the QUADSPI starts an operation with the flash memory, the BUSY bit is automatically set in QUADSPI_SR.

- In indirect mode, BUSY is reset once the QUADSPI has completed the requested command sequence, and the FIFO is empty.
- In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1, or due to an abort.
- After the first access in memory-mapped mode, BUSY goes low only on a timeout event or on an abort.

Any operation can be aborted by setting the ABORT bit in QUADSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

*Note:* Some flash memories may misbehave if a write operation to a status registers is aborted.

15.3.15 NCS behavior

By default, NCS is high, deselecting the external flash memory. NCS falls before an operation begins and rises as soon as it finishes.
When CKMODE = 0 ("mode0", where CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

**Figure 36. NCS when CKMODE = 0 (T = CLK period)**

When CKMODE=1 ("mode3", where CLK goes high when no operation is in progress) and DDRM=0 (SDR mode), NCS still falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

**Figure 37. NCS when CKMODE = 1 in SDR mode (T = CLK period)**

When CKMODE = 1 ("mode3") and DDRM = 1 (DDR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge, as shown in the figure below. Because DDR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

**Figure 38. NCS when CKMODE = 1 in DDR mode (T = CLK period)**
When the FIFO stays full in a read operation or if the FIFO stays empty in a write operation, the operation stalls and CLK stays low until firmware services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later, as shown in Figure 39.

Figure 39. NCS when CKMODE = 1 with an abort (T = CLK period)

15.4 QUADSPI interrupts

An interrupt can be produced on the following events:
- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Table 71. QUADSPI interrupt requests

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>TOF</td>
<td>TOIE</td>
</tr>
<tr>
<td>Status match</td>
<td>SMF</td>
<td>SMIE</td>
</tr>
<tr>
<td>FIFO threshold</td>
<td>FTF</td>
<td>FTIE</td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TCF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TEF</td>
<td>TEIE</td>
</tr>
</tbody>
</table>
15.5 QUADSPI registers

15.5.1 QUADSPI control register (QUADSPI_CR)

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>PRESCALER[7:0]</td>
</tr>
<tr>
<td>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>rw      rw      rw      rw      rw      rw      rw</td>
</tr>
<tr>
<td>rw      rw      rw      rw      rw      rw      rw</td>
</tr>
</tbody>
</table>

Bits 31:24 PRESCALER[7:0]: Clock prescaler
This bitfield defines the scaler factor for generating CLK based on the clock (value+1).
0: FCLK = F, clock used directly as QUADSPI CLK (prescaler bypassed)
1: FCLK = F/2
2: FCLK = F/3
...
255: FCLK = F/256
For odd clock division factors, CLK duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.
When setting quad-SPI interface in DDR mode, the prescaler must be set with a division ratio of two or more.

Note: This bitfield can be modified only when BUSY = 0.

Bit 23 PMM: Polling match mode
This bit indicates which method must be used for determining a “match” during automatic status-polling mode.
0: AND match mode. SMF is set if all the unmasked bits received from the flash memory match the corresponding bits in the match register.
1: OR match mode. SMF is set if any one of the unmasked bits received from the flash memory matches its corresponding bit in the match register.

Note: This bit can be modified only when BUSY = 0.

Bit 22 APMS: Automatic status-polling mode stop
This bit determines if automatic status-polling is stopped after a match.
0: Automatic status-polling mode is stopped only by abort or by disabling the QUADSPI.
1: Automatic status-polling mode stops as soon as there is a match.

Note: This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 TOIE: Timeout interrupt enable
This bit enables the timeout interrupt.
0: Interrupt disabled
1: Interrupt enabled
Bit 19 **SMIE**: Status match interrupt enable
- This bit enables the status match interrupt.
  - 0: Interrupt disabled
  - 1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
- This bit enables the FIFO threshold interrupt.
  - 0: Interrupt disabled
  - 1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
- This bit enables the transfer complete interrupt.
  - 0: Interrupt disabled
  - 1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
- This bit enables the transfer error interrupt.
  - 0: Interrupt disabled
  - 1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level
- This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag (bit FTF in register QUADSPI_SR) to be set.
  - 0: In indirect-write mode (FMODE = 00), FTF is set if there are one or more free bytes location left in the FIFO or indirect-read mode (FMODE = 01), FTF is set if there are one or more valid bytes that can be read from the FIFO.
  - 1: In indirect-write mode (FMODE = 00), FTF is set if there are two or more free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are two or more valid bytes that can be read from the FIFO
  ... 31: In indirect-write mode (FMODE = 00), FTF is set if there are 32 free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are 32 valid bytes that can be read from the FIFO.
- If DMAEN = 1, then the DMA controller for the corresponding channel must be disabled before changing the FTHRES value.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SSHIFT**: Sample shift
- By default, the QUADSPI samples data 1/2 of a CLK cycle after the data is driven by the flash memory. This bit allows the data to be sampled later in order to account for external signal delays.
  - 0: No shift
  - 1: 1/2 cycle shift
- The firmware must assure that SSHIFT = 0 when in DDR mode (when DDRM = 1).

*Note*: This bitfield can be modified only when BUSY = 0.
Bit 3  **TCEN**: Timeout counter enable

This bit is valid only when memory-mapped mode (FMODE = 11) is selected. Activating this bit causes the NCS to be released (and thus reduces consumption) if there has not been an access after a certain amount of time, where this time is defined by TIMEOUT[15:0] (QUADSPI_LPTR). This bit enables the timeout counter.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without an access since when the FIFO becomes full with prefetch data.

0: Timeout counter is disabled, and thus the NCS remains active indefinitely after an access in memory-mapped mode.
1: Timeout counter is enabled, and thus the NCS is released in memory-mapped mode after TIMEOUT[15:0] cycles of flash memory inactivity.

*Note*: This bit can be modified only when BUSY = 0.

Bit 2  **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via the QUADSPI_DR register. DMA transfers are initiated when the FIFO threshold flag, FTF, is set.

0: DMA is disabled for indirect mode.
1: DMA is enabled for indirect mode.

Bit 1  **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is complete. This bit stops the current transfer.

In automatic status-polling or memory-mapped mode, this bit also reset APM or DM bit.

0: No abort requested
1: Abort requested

Bit 0  **EN**: QUADSPI enable

0: QUADSPI disabled
1: QUADSPI enabled

### 15.5.2 QUADSPI device configuration register (QUADSPI_DCR)

Address offset: 0x004

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 31:21  Reserved, must be kept at reset value.
Bits 20:16 **FSIZE[4:0]**: Flash memory size
This bitfield defines the size of external memory using the following formula:
Number of bytes in flash memory = 2^{FSIZE+1}
FSIZE+1 is effectively the number of address bits required to address the flash memory.
The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.
*Note:* This bitfield can be modified only when BUSY = 0.

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

Bits 10:8 **CSHT[2:0]**: Chip select high time
CSHT+1 defines the minimum number of CLK cycles which the chip select (NCS) must remain high between commands issued to the flash memory.
0: NCS stays high for at least 1 cycle between flash memory commands
1: NCS stays high for at least 2 cycles between flash memory commands
...
7: NCS stays high for at least 8 cycles between flash memory commands
*Note:* This bitfield can be modified only when BUSY = 0.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **CKMODE**: Mode 0/mode 3
This bit indicates the level that CLK takes between commands (when NCS = 1).
0: CLK must stay low while NCS is high (chip select released). This is referred to as mode 0.
1: CLK must stay high while NCS is high (chip select released). This is referred to as mode 3.
*Note:* This bitfield can be modified only when BUSY = 0.

---

**15.5.3 QUADSPI status register (QUADSPI_SR)**

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>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>0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level
This bitfield gives the number of valid bytes which are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full. In memory-mapped mode and in automatic status-polling mode, FLEVEL is zero.

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

Bit 5 **BUSY**: Busy
This bit is set when an operation is on going. This bit clears automatically when the operation with the flash memory is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag
This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.
Bit 3 **SMF**: Status match flag
This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (QUADSPI_PSMAR). It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag
In indirect mode, this bit is set when the FIFO threshold is reached, or if there is any data left in the FIFO after reads from the flash memory are complete. It is cleared automatically as soon as threshold condition is no longer true.

In automatic status-polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag
This bit is set in indirect mode when the programmed number of data is transferred or in any mode when the transfer is aborted. It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag
This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

### 15.5.4 QUADSPI flag clear register (QUADSPI_FCR)

Address offset: 0x00C
 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 31:5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in QUADSPI_SR.

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in QUADSPI_SR.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in QUADSPI_SR.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in QUADSPI_SR.
15.5.5 QUADSPI data length register (QUADSPI_DLR)

Address offset: 0x010  
Reset value: 0x0000 0000

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

Bits 31:0 DL[31:0]: Data length
Number of data to be retrieved (value+1) in indirect and automatic status-polling modes.
A value no greater than 3 (indicating 4 bytes) must be used for automatic status-polling mode.
All 1s in indirect mode means undefined length, where the QUADSPI continues until the end of memory, as defined by FSIZE.
0x0000_0000: 1 byte is to be transferred
0x0000_0001: 2 bytes are to be transferred
0x0000_0002: 3 bytes are to be transferred
0x0000_0003: 4 bytes are to be transferred
... 
0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred
0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred
0xFFFF_FFFF: undefined length -- all bytes until the end of flash memory (as defined by FSIZE) are to be transferred. Continue reading indefinitely if FSIZE = 0x1F.
This bitfield has no effect when in memory-mapped mode (FMODE = 10).

Note: This bitfield can be written only when BUSY = 0.

15.5.6 QUADSPI communication configuration register (QUADSPI_CCR)

Address offset: 0x014
Reset value: 0x0000 0000

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

Bit 31 DDRM: Double data rate mode
This bit sets the DDR mode for the address, alternate byte and data phase:
0: DDR mode disabled
1: DDR mode enabled

Note: This bit can be written only when BUSY = 0.

Bit 30 Reserved, must be kept at reset value.
Bit 29 Reserved, must be kept at reset value.
Bit 28 **SIOO**: Send instruction only once mode  
This bit has no effect when IMODE = 00. See Section 15.3.12 for more details.  
0: Instruction sent on every transaction  
1: Instruction sent only for the first command  
*Note:* This bit can be written only when BUSY = 0.

Bits 27:26 **FMODE[1:0]**: Functional mode  
This bitfield defines the QUADSPI functional mode of operation.  
00: Indirect-write mode  
01: Indirect-read mode  
10: Automatic status-polling mode  
11: Memory-mapped mode  
If DMAEN = 1 already, the DMA controller for the corresponding channel must be disabled before changing the FMODE value.  
*Note:* This bitfield can be written only when BUSY = 0.

Bits 25:24 **DMODE[1:0]**: Data mode  
This bitfield defines the data phase mode of operation:  
00: No data  
01: Data on a single line  
10: Data on two lines  
11: Data on four lines  
This bitfield also determines the dummy phase mode of operation.  
*Note:* This bitfield can be written only when BUSY = 0.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **DCYC[4:0]**: Number of dummy cycles  
This bitfield defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of CLK cycles (0-31).  
*Note:* This bitfield can be written only when BUSY = 0.

Bits 17:16 **ABSIZE[1:0]**: Alternate-byte size  
This bit defines the size of alternate bytes.  
00: 8-bit alternate byte  
01: 16-bit alternate byte  
10: 24-bit alternate bytes  
11: 32-bit alternate bytes  
*Note:* This bitfield can be written only when BUSY = 0.

Bits 15:14 **ABMODE[1:0]**: Alternate byte mode  
This bitfield defines the alternate-byte phase mode of operation.  
00: No alternate bytes  
01: Alternate bytes on a single line  
10: Alternate bytes on two lines  
11: Alternate bytes on four lines  
*Note:* This bitfield can be written only when BUSY = 0.

Bits 13:12 **ADSIZE[1:0]**: Address size  
This bit defines address size:  
00: 8-bit address  
01: 16-bit address  
10: 24-bit address  
11: 32-bit address  
*Note:* This bitfield can be written only when BUSY = 0.
15.5.7 QUADSPI address register (QUADSPI_AR)

Address offset: 0x018
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>ADDRESS[31:0]</th>
<th>Address to be sent to the external flash memory. Writes to this bitfield are ignored when BUSY = 1 or when FMODE = 11 (memory-mapped mode).</th>
</tr>
</thead>
</table>

15.5.8 QUADSPI alternate-byte register (QUADSPI_ABR)

Address offset: 0x01C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>ALTERNATE[31:0]</th>
<th>Address mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:10</td>
<td>ADMODE[1:0]</td>
<td>Address mode for the external SPI device. This bitfield defines the address phase mode of operation.</td>
</tr>
<tr>
<td>9:8</td>
<td>IMODE[1:0]</td>
<td>Instruction mode for the external SPI device. This bitfield defines the instruction phase mode of operation.</td>
</tr>
<tr>
<td>7:0</td>
<td>INSTRUCTION[7:0]</td>
<td>Instruction to be sent to the external SPI device.</td>
</tr>
<tr>
<td>31:16</td>
<td>ADDRESS[31:16]</td>
<td>Address to be sent to the external flash memory.</td>
</tr>
<tr>
<td>15:0</td>
<td>ADDRESS[15:0]</td>
<td>Address to be sent to the external flash memory.</td>
</tr>
</tbody>
</table>
Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be send to the external SPI device right after the address.

*Note: This bitfield can be written only when BUSY = 0.*

### 15.5.9 QUADSPI data register (QUADSPI_DR)

Address offset: 0x020

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>DATA[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>
<td></td>
</tr>
<tr>
<td>DATA[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>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device.

In indirect write mode, data written to this register is stored on the FIFO before it is sent to the flash memory during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect read mode, reading this register gives (via the FIFO) the data which was received from the flash memory. If the FIFO does not have as many bytes as requested by the read operation and if BUSY=1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the flash memory (without masking).

Word, halfword, and byte accesses to this register are supported. In indirect write mode, a byte write adds 1 byte to the FIFO, a halfword write 2, and a word write 4. Similarly, in indirect read mode, a byte read removes 1 byte from the FIFO, a halfword read 2, and a word read 4. Accesses in indirect mode must be aligned to the bottom of this register: a byte read must read DATA[7:0] and a halfword read must read DATA[15:0].

### 15.5.10 QUADSPI polling status mask register (QUADSPI_PSMKR)

Address offset: 0x024

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>MASK[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>
<td></td>
</tr>
<tr>
<td>MASK[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>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:0  **MASK[31:0]**: Status mask
Mask to be applied to the status bytes received in automatic status-polling mode.
For bit n:
0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic
1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic

*Note: This bitfield can be written only when BUSY = 0.*

### 15.5.11 QUADSPI polling status match register (QUADSPI_PSMAR)

**Address offset:** 0x028

**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>

**MATCH[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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>

**MATCH[15:0]**

Bits 31:0  **MATCH[31:0]**: Status match
Value to be compared with the masked status register to get a match.

*Note: This bitfield can be written only when BUSY = 0.*

### 15.5.12 QUADSPI polling interval register (QUADSPI_PIR)

**Address offset:** 0x02C

**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>

**INTERVAL[15:0]**

<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>
<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  **INTERVAL[15:0]**: Polling interval
Number of CLK cycles between two read during automatic status-polling phases.

*Note: This bitfield can be written only when BUSY = 0.*
15.5.13 QUADSPI low-power timeout register (QUADSPI_LPTR)

Address offset: 0x030
Reset value: 0x0000 0000

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

Bits 15:0 \textit{TIMEOUT}[15:0]: Timeout period

After each access in memory-mapped mode, the QUADSPI prefetches the subsequent bytes and holds these bytes in the FIFO. This bitfield indicates how many CLK cycles the QUADSPI waits after the FIFO becomes full until it raises NCS, putting the flash memory in a lower-consumption state.

\textit{Note: This bitfield can be written only when BUSY = 0.}

15.5.14 QUADSPI register map

Table 72. QUADSPI 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 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | QUADSPI_CR    | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| QUADSPI_DCR |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| QUADSPI_SR  |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| QUADSPI_FCR  |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| QUADSPI_DLR  |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| QUADSPI_CCR  |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| QUADSPI_AR   |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| QUADSPI_ABR  |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| QUADSPI_DR   |            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.2 for the register boundary addresses.
16 Analog-to-digital converter (ADC)

16.1 Introduction

The ADC consists of a 12-bit successive approximation analog-to-digital converter. The ADC has up to 19 multiplexed channels. 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 ADC is mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performance while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

16.2 ADC main features

- High-performance features
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - 4 dedicated data registers for the injected channels

- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256
  - Programmable data shift up to 8-bit

- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency
application (auto-delayed mode)

Number of external analog input channels
– Up to 5 fast channels from GPIO pads
– Up to 11 slow channels from GPIO pads

In addition, there are several internal dedicated channels
– The internal reference voltage (V_{REFINT}), connected to ADC1
– The internal temperature sensor (V_{TS}), connected to ADC1
– The V_{BAT} monitoring channel (V_{BAT}/3), connected to ADC1

Start-of-conversion can be initiated:
– By software for both regular and injected conversions
– By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions

Conversion modes
– The ADC 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 ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
3 analog watchdogs

ADC input range: V_{REF-} ≤ V_{IN} ≤ V_{REF+}

Figure 40 shows the block diagram of one ADC.

### 16.3 ADC implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC1_INN</td>
<td>1 to 15</td>
<td>5 to 12 and 15</td>
</tr>
<tr>
<td>ADC1_INP</td>
<td>1 to 16</td>
<td>5 to 13, 15 to 16</td>
</tr>
</tbody>
</table>
16.4 ADC functional description

16.4.1 ADC block diagram

*Figure 40* shows the ADC block diagram and *Table 75* gives the ADC pin description.

*Figure 40. ADC block diagram*
### 16.4.2 ADC pins and internal signals

#### Table 74. 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>EXT[15:0]</td>
<td>Inputs</td>
<td>Up to 16 external trigger inputs for the regular conversions (can be connected to on-chip timers).</td>
</tr>
<tr>
<td>JEXT[15:0]</td>
<td>Inputs</td>
<td>Up to 16 external trigger inputs for the injected conversions (can be connected to on-chip timers).</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>
<tr>
<td>V_TS</td>
<td>Input</td>
<td>Output voltage from internal temperature sensor</td>
</tr>
<tr>
<td>VREFINT</td>
<td>Input</td>
<td>Output voltage from internal reference voltage</td>
</tr>
<tr>
<td>V_BAT</td>
<td>Input supply</td>
<td>External battery voltage supply</td>
</tr>
</tbody>
</table>

#### Table 75. ADC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREF+</td>
<td>Input, analog reference positive</td>
<td>The higher/positive reference voltage for the ADC</td>
</tr>
<tr>
<td>VDDA</td>
<td>Input, analog supply</td>
<td>Analog power supply equal VDDA</td>
</tr>
<tr>
<td>VREF−</td>
<td>Input, analog reference negative</td>
<td>The lower/negative reference voltage for the ADC. VREF− is internally connected to VSSA</td>
</tr>
<tr>
<td>VSSA</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply. On device package which do not have a dedicated VSSA pin, VSSA is internally connected to VSS.</td>
</tr>
<tr>
<td>VPin</td>
<td>Positive analog input channels</td>
<td>Connected either to ADC1_INPi external channels or to internal channels. This input is converted in single-ended mode</td>
</tr>
<tr>
<td>VInNi</td>
<td>Negative analog input channels</td>
<td>Connected either to VREF− or to external channels: ADC1_INNi and ADC1_INP[i+1].</td>
</tr>
<tr>
<td>ADC1_INNi</td>
<td>Negative external analog input signals</td>
<td>Up to 15 analog input channels Refer to Section 16.4.4: ADC1 connectivity for details.</td>
</tr>
<tr>
<td>ADC1_INPi</td>
<td>Positive external analog input signals</td>
<td>Up to 16 analog input channels Refer to Section 16.4.4: ADC1 connectivity for details</td>
</tr>
</tbody>
</table>
16.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The input clock is and can be selected between two different clock sources (see Figure 41: ADC clock scheme):

1. The ADC clock can be a specific clock source, derived from the following clock sources:
   - The system clock
   - PLLSAI1 (single ADC implementation)
   Refer to RCC Section for more information on how to generate ADC dedicated clock.
   To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be reset.

2. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).
   To select this scheme, bits CKMODE[1:0] of the ADCx_CCR register must be different from 00.

Note: For option 2), a prescaling factor of 1 (CKMODE[1:0] = 01) can be used only if the AHB prescaler is set (HPRE[3:0] = 0xx in RCC_CFGR register).

Option 1) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits PRESC[3:0] in the ADCx_CCR register.

Option 2) 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 time is added by the resynchronizations between the two clock domains).
Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- \( F_{\text{HCLK}} \geq F_{\text{ADC}} / 4 \) if the resolution of all channels are 12-bit or 10-bit
- \( F_{\text{HCLK}} \geq F_{\text{ADC}} / 3 \) if there are some channels with resolutions equal to 8-bit (and none with lower resolution)
- \( F_{\text{HCLK}} \geq F_{\text{ADC}} / 2 \) if there are some channels with resolutions equal to 6-bit
16.4.4 ADC1 connectivity

Figure 42. ADC1 connectivity
16.4.5 Slave AHB interface

The ADC implements an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:
- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

16.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, follow the sequence below:
1. Exit Deep-power-down mode by clearing DEEPPWD bit.
2. Enable the ADC voltage regulator by setting ADVREGEN.
3. Wait for the startup time to configure the ADC (refer to the device datasheet for the value of the startup time).

When ADC operations are complete, the ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD = 1 into ADC_CR register. This is particularly interesting before entering Stop mode.

Note: Writing DEEPPWD = 1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to Section 16.4.8: Calibration (ADCAL, ADCALDIFF, ADC_CALFACT)).

Before entering Stop 2 mode, the ADC needs to be disabled by setting ADDIS and clearing ADVREGEN.
16.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy_INPx external voltage equal to VINP[i] (positive input) and VREF− (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the ADCy_INPx external voltage positive input equal to VINP[i], and the ADCy_INNx negative input equal to VINN[i].

The input voltage in differential mode ranges from VREF− to VREF+, which makes a full scale range of 2xVREF+. When VINP[i] equals VREF−, VINN[i] equals VREF−, and the maximum negative input differential voltage (VREF−) corresponds to 0x000 ADC output. When VINP[i] equals VREF+, VINN[i] equals VREF+, and the maximum positive input differential voltage (VREF+) corresponds to 0xFFF ADC output. When VINP[i] and VINN[i] are connected together, the zero input differential voltage corresponds to 0x800 ADC output.

The ADC sensitivity in differential mode is twice smaller than in single-ended mode.

When ADC is configured as differential mode, both inputs should be biased at (VREF+) / 2 voltage. Refer to the device datasheet for the allowed common mode input voltage VCMIN.

The input signals are supposed to be differential (common mode voltage should be fixed). Internal channels (such as VTS and VREFINT) are used in single-ended mode only.

For a complete description of how the input channels are connected, refer to Section 16.4.4: ADC1 connectivity.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage VINN[i] is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted.

16.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

The ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration to be applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration to be applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when 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. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration).

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when VREF+ voltage changed more than 10%.

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC_CALFACT register.

**Figure 43. ADC calibration**

<table>
<thead>
<tr>
<th>ADCALDIF</th>
<th>0: Single-ended input</th>
<th>1: Differential input</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCAL</td>
<td></td>
<td>tCAL</td>
</tr>
<tr>
<td>ADC State</td>
<td>OFF</td>
<td>Startup</td>
</tr>
<tr>
<td>CALFACT_x[6:0]</td>
<td>0x00</td>
<td>Calibration factor</td>
</tr>
</tbody>
</table>

by S/W  by H/W  Indicative timings
Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 44. Updating the ADC calibration factor

Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.
16.4.9 **ADC on-off control (ADEN, ADDIS, ADRDY)**

First of all, follow the procedure explained in Section 16.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)).

Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of $t_{\text{STAB}}$ before it starts converting accurately, as shown in Figure 46. Two control bits enable or disable the ADC:

- ADEN = 1 enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 (refer to Section 16.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART = 1 or when an external injected trigger event occurs, if injected triggers are enabled.

**Software procedure to enable the ADC**

1. Clear the ADRDY bit in the ADC_ISR register by writing 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRYD is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing 1 (optional).

**Caution:** ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).
**Software procedure to disable the ADC**

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.

2. Set ADDIS.

3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

**16.4.10 Constraints when writing the ADC control bits**

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).

- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN = 1). Refer to Section 16.7.16: ADC injected sequence register (ADC_JSQR) for additional details.
Note: **There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN as well as all the bits of ADC_CR register).**

### 16.4.11 Channel selection (SQRx, JSQRx)

There are up to 19 multiplexed channels:

- Up to 11 slow analog inputs coming from GPIO pads (ADC1_INP/INN[6:16])
  - Depending on the products, not all of them are available on GPIO pads.
  - The ADC is connected to the following internal analog inputs:
    - The internal reference voltage (VREFINT) is connected to ADC1_INP0.
    - The internal temperature sensor (VTS) is connected to ADC1_INP17.
    - The VBAT monitoring channel (VBAT/3) is connected to ADC1_INP18.

**Note:** To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, CH17SEL or CH18SEL in the ADCx_CCR registers.

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC1_INP/INN3, ADC1_INP/INN8, ADC1_INP/INN2, ADC1_INN/INP2, ADC1_INP/INN0, ADC1_INP/INN2, ADC1_INP/INN15.

A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.

An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to **Section 16.4.17: Stopping an ongoing conversion (ADSTP, JADSTP)**).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC_CFGR register). Refer to **Section 16.4.21: Queue of context for injected conversions**.
16.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

$$T_{CONV} = \text{Sampling time} + 12.5 \text{ ADC clock cycles}$$

Example:

With $F_{ADC\_CLK} = 30$ MHz and a sampling time of 2.5 ADC clock cycles:

$$T_{CONV} = (2.5 + 12.5) \text{ ADC clock cycles} = 15 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

I/O analog switch voltage booster

The resistance of the I/O analog switches increases when the VDDA voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low VDDA voltage by enabling an internal voltage booster through the BOOSTEN bit of the SYSCFG_CFGR1 register.

16.4.13 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)
Inside the regular sequence, after each conversion is complete:
   - The converted data are stored into the 16-bit ADC_DR register
   - The EOC (end of regular conversion) flag is set
   - An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:
   - The converted data are stored into one of the four 16-bit ADC_JDRy registers
   - The JEOC (end of injected conversion) flag is set
   - An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:
   - The EOS (end of regular sequence) flag is set
   - An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:
   - The JEOS (end of injected sequence) flag is set
   - An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

16.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversion of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:
   - The converted data are stored into 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 DISCEN = 1 and CONT = 1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section).
16.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:
- Immediately: if EXTEN[1:0] = 00 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN[1:0] is not equal to 00

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:
- Immediately, if JEXTEN[1:0] = 00 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN[1:0] is not equal to 00

Note: In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:
- In single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT=x, EXTSEL=x)
  - After execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTSEL≠0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:
- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- In all cases (JEXTSEL=x)
  - After execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.
16.4.16 ADC timing

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:

\[
T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = [2.5 \text{ min} + 12.5 \text{ 12bit}] \times T_{\text{ADC_CLK}}
\]

\[
T_{\text{CONV}} = T_{\text{SMPL}} + T_{\text{SAR}} = 83.33 \text{ ns min} + 416.67 \text{ ns 12bit} = 500.0 \text{ ns (for } F_{\text{ADC_CLK}} = 30 \text{ MHz)}
\]

Figure 47. Analog to digital conversion time

1. \(T_{\text{SMPL}}\) depends on \(SMP[2:0]\).
2. \(T_{\text{SAR}}\) depends on \(RES[2:0]\).

16.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (e.g. changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.
**Note:** In auto-injection mode \((\text{JAUTO} = 1)\), setting ADSTP bit aborts both regular and injected conversions \((\text{JADSTP} \text{ must not be used})\).

### Figure 48. Stopping ongoing regular conversions

<table>
<thead>
<tr>
<th>ADC state</th>
<th>Trigger</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDY</td>
<td>Sample Ch(N-1)</td>
<td>Convert Ch(N-1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JADSTART</th>
<th>Cleared by HW</th>
<th>REGULAR CONVERSIONS ongoing</th>
<th>Cleared by HW</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADSTART</td>
<td>Cleared by HW</td>
<td>(software is not allowed to configure regular conversions selection and triggers)</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADSTP</td>
<td>Cleared by SW</td>
<td>Set by SW</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADC_DR</td>
<td>Data N-2</td>
<td>Data N-1</td>
<td></td>
</tr>
</tbody>
</table>

### Figure 49. Stopping ongoing regular and injected conversions

<table>
<thead>
<tr>
<th>ADC state</th>
<th>Regular trigger</th>
<th>Injected trigger</th>
<th>Regular trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDY</td>
<td>Sample Ch(N-1)</td>
<td>Convert Ch(N-1)</td>
<td>RDY</td>
</tr>
<tr>
<td>JADSTART</td>
<td>Set by SW</td>
<td>INJECTED CONVERSIONS ongoing</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>JADSTP</td>
<td>Cleared by HW</td>
<td>Set by SW</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADC_JDR</td>
<td>DATA M-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADSTART</td>
<td>Set by SW</td>
<td>REGULAR CONVERSIONS ongoing</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADSTP</td>
<td>Cleared by SW</td>
<td>Set by SW</td>
<td>Cleared by HW</td>
</tr>
<tr>
<td>ADC_DR</td>
<td>DATA N-2</td>
<td>DATA N-1</td>
<td></td>
</tr>
</tbody>
</table>

**MSv30533V2**

**MS30534V1**
16.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g., timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

Table 76 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

<table>
<thead>
<tr>
<th>EXTEN[1:0]</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Hardware Trigger detection disabled, software trigger detection enabled</td>
</tr>
<tr>
<td>01</td>
<td>Hardware Trigger with detection on the rising edge</td>
</tr>
<tr>
<td>10</td>
<td>Hardware Trigger with detection on the falling edge</td>
</tr>
<tr>
<td>11</td>
<td>Hardware Trigger with detection on both the rising and falling edges</td>
</tr>
</tbody>
</table>

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 77. Configuring the trigger polarity for injected external triggers

<table>
<thead>
<tr>
<th>JEXTEN[1:0]</th>
<th>Source</th>
</tr>
</thead>
</table>
| 00          | – If JQDIS = 1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled  
|             | – If JQDIS = 0 (Queue enabled), Hardware and software trigger detection disabled |  
| 01          | Hardware Trigger with detection on the rising edge |  
| 10          | Hardware Trigger with detection on the falling edge |  
| 11          | Hardware Trigger with detection on both the rising and falling edges |  

Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to Section 16.4.21: Queue of context for injected conversions.

The EXTSEL and JEXTSEL control bits select which out of 16 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.
Note: The regular trigger selection cannot be changed on-the-fly. The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 16.4.21: Queue of context for injected conversions on page 442.

Table 78 to Table 79 give all the possible external triggers of the three ADCs for regular and injected conversions.

### Table 78. ADC1 - External triggers for regular channels

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Type</th>
<th>EXTSEL[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXT0</td>
<td>TIM1_CC1</td>
<td>Internal signal from on-chip timers</td>
<td>0000</td>
</tr>
<tr>
<td>EXT1</td>
<td>TIM1_CC2</td>
<td>Internal signal from on-chip timers</td>
<td>0001</td>
</tr>
<tr>
<td>EXT2</td>
<td>TIM1_CC3</td>
<td>Internal signal from on-chip timers</td>
<td>0010</td>
</tr>
<tr>
<td>EXT3</td>
<td>TIM2_CC2</td>
<td>Internal signal from on-chip timers</td>
<td>0011</td>
</tr>
<tr>
<td>EXT4</td>
<td>-</td>
<td>-</td>
<td>0100</td>
</tr>
<tr>
<td>EXT5</td>
<td>-</td>
<td>-</td>
<td>0101</td>
</tr>
<tr>
<td>EXT6</td>
<td>EXTI Line 11</td>
<td>External pin</td>
<td>0110</td>
</tr>
<tr>
<td>EXT9</td>
<td>TIM1_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>1001</td>
</tr>
<tr>
<td>EXT10</td>
<td>TIM1_TRGO2</td>
<td>Internal signal from on-chip timers</td>
<td>1010</td>
</tr>
<tr>
<td>EXT11</td>
<td>TIM2_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>1011</td>
</tr>
<tr>
<td>EXT13</td>
<td>-</td>
<td>-</td>
<td>1101</td>
</tr>
<tr>
<td>EXT14</td>
<td>-</td>
<td>-</td>
<td>1110</td>
</tr>
</tbody>
</table>

### Table 79. ADC1 - External trigger for injected channels

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Type</th>
<th>JEXTSEL[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEXT0</td>
<td>TIM1_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>0000</td>
</tr>
<tr>
<td>JEXT1</td>
<td>TIM1_CC4</td>
<td>Internal signal from on-chip timers</td>
<td>0001</td>
</tr>
<tr>
<td>JEXT2</td>
<td>TIM2_TRGO</td>
<td>Internal signal from on-chip timers</td>
<td>0010</td>
</tr>
<tr>
<td>JEXT3</td>
<td>TIM2_CH1</td>
<td>Internal signal from on-chip timers</td>
<td>0011</td>
</tr>
<tr>
<td>JEXT4</td>
<td>-</td>
<td>-</td>
<td>0100</td>
</tr>
<tr>
<td>JEXT5</td>
<td>-</td>
<td>-</td>
<td>0101</td>
</tr>
<tr>
<td>JEXT6</td>
<td>EXTI Line 15</td>
<td>External pin</td>
<td>0110</td>
</tr>
<tr>
<td>JEXT8</td>
<td>TIM1_TRGO2</td>
<td>Internal signal from on-chip timers</td>
<td>1000</td>
</tr>
<tr>
<td>JEXT14</td>
<td>-</td>
<td>-</td>
<td>1110</td>
</tr>
<tr>
<td>JEXT15</td>
<td>-</td>
<td>-</td>
<td>1111</td>
</tr>
</tbody>
</table>
16.4.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).

3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.

4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. Figure 50 shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously. When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence is stopped upon DMA Transfer Complete event.
16.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.
Analog-to-digital converter (ADC)

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:
- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

16.4.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:
- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)
All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

The JSQR register can be written at any moment even when injected conversions are ongoing.

Each data written into the JSQR register is stored into the Queue of context.

At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.

Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.

A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.

Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:

- If JQM = 0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
- If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.

Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.

The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling the ADC by setting ADDIS = 1:

- If JQM = 0, the Queue is maintained with the last active context.
- If JQM = 1, the Queue becomes empty and triggers are ignored.

Note: When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue. The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out
2nd trigger, discontinuous. Sequence 1: 2nd conversion.
3rd trigger, discontinuous. Sequence 1: 3rd conversion.
4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.
5th trigger, discontinuous. Sequence 2: 2nd conversion.
6th trigger, discontinuous. Sequence 2: 3rd conversion.
Behavior when changing the trigger or sequence context

The Figure 51 and Figure 52 show the behavior of the context Queue when changing the sequence or the triggers.

**Figure 51. Example of JSQR queue of context (sequence change)**

1. Parameters:
   - P1: sequence of 3 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 1
   - P3: sequence of 4 conversions, hardware trigger 1

**Figure 52. Example of JSQR queue of context (trigger change)**

1. Parameters:
   - P1: sequence of 2 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 2
   - P3: sequence of 4 conversions, hardware trigger 1
Queue of context: Behavior when a queue overflow occurs

The Figure 53 and Figure 54 show the behavior of the context Queue if an overflow occurs before or during a conversion.

**Figure 53. Example of JSQR queue of context with overflow before conversion**

1. Parameters:
   - P1: sequence of 2 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 2
   - P3: sequence of 3 conversions, hardware trigger 1
   - P4: sequence of 4 conversions, hardware trigger 1

**Figure 54. Example of JSQR queue of context with overflow during conversion**

1. Parameters:
   - P1: sequence of 2 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 2
   - P3: sequence of 3 conversions, hardware trigger 1
   - P4: sequence of 4 conversions, hardware trigger 1
It is recommended to manage the queue overflows as described below:

After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).

Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

**Queue of context: Behavior when the queue becomes empty**

*Figure 55* and *Figure 56* show the behavior of the context Queue when the Queue becomes empty in both cases JQM = 0 or 1.

**Figure 55. Example of JSQR queue of context with empty queue (case JQM = 0)**

1. Parameters:
   
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

**Note:**  When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.
Figure 56. Example of JSQR queue of context with empty queue (case JQM = 1)

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 57. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0)
Case when JADSTP occurs during an ongoing conversion

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Figure 58. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0)
Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 59. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0)
Case when JADSTP occurs outside an ongoing conversion

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Figure 60. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1)

1. Parameters:
   - P1: sequence of 1 conversion, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 1
   - P3: sequence of 1 conversion, hardware trigger 1

Figure 61. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0)

1. Parameters:
   - P1: sequence of 1 conversion, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 1
   - P3: sequence of 1 conversion, hardware trigger 1
Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN[1:0] not equal to 00 (otherwise triggering a software conversion).
6. Set JADSTART.
7. Set JADSTP.
8. Wait until JADSTART is reset.
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC_CFGR register.

16.4.22 Programmable resolution (RES) - Fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. Figure 67, Figure 68, Figure 69 and Figure 70 show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to Table 80.
16.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

16.4.24 End of conversion sequence (EOS, JEOS)

The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

### Table 80. $T_{SAR}$ timings depending on resolution

<table>
<thead>
<tr>
<th>RES (bits)</th>
<th>$T_{SAR}$ (ADC clock cycles)</th>
<th>$T_{SAR}$ (ns) at $F_{ADC} = 30$ MHz</th>
<th>$T_{CONV}$ (ADC clock cycles) (with Sampling Time= 2.5 ADC clock cycles)</th>
<th>$T_{CONV}$ (ns) at $F_{ADC} = 30$ MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12.5 ADC clock cycles</td>
<td>416.67 ns</td>
<td>15 ADC clock cycles</td>
<td>500.0 ns</td>
</tr>
<tr>
<td>10</td>
<td>10.5 ADC clock cycles</td>
<td>350.0 ns</td>
<td>13 ADC clock cycles</td>
<td>433.33 ns</td>
</tr>
<tr>
<td>8</td>
<td>8.5 ADC clock cycles</td>
<td>203.33 ns</td>
<td>11 ADC clock cycles</td>
<td>366.67 ns</td>
</tr>
<tr>
<td>6</td>
<td>6.5 ADC clock cycles</td>
<td>216.67 ns</td>
<td>9 ADC clock cycles</td>
<td>300.0 ns</td>
</tr>
</tbody>
</table>
16.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

**Figure 63. Single conversions of a sequence, software trigger**

1. EXTEN[1:0] = 00, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

**Figure 64. Continuous conversion of a sequence, software trigger**

1. EXTEN[1:0] = 00, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.
Figure 65. Single conversions of a sequence, hardware trigger

1. TRGx (over-frequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 66. Continuous conversions of a sequence, hardware trigger

1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.
16.4.26 Data management

Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CHAN, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the
converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the
converted data is stored into the corresponding ADC_JDRy data register which is 16 bits
wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after
conversion. Data can be right- or left-aligned as shown in Figure 67, Figure 68, Figure 69
and Figure 70.

Special case: when left-aligned, the data are aligned on a half-word basis except when the
resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in
Figure 69 and Figure 70.

Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is
set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.

Offset

An offset y (y = 1, 2, 3, 4) can be applied to a channel by setting the bit OFFSETy_EN = 1 into
ADC_OFRy register. The channel to which the offset is applied is programmed into the bits
OFFSETy_CHAN[4:0] of ADC_OFRy register. In this case, the converted value is decreased by
the user-defined offset written in the bits OFFSETy[11:0]. The result may be a negative
value so the read data is signed and the SEXT bit represents the extended sign value.

Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is
set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as
reset).

Table 83 describes how the comparison is performed for all the possible resolutions for
analog watchdog 1.

<table>
<thead>
<tr>
<th>Resolution (bits RES[1:0])</th>
<th>Subtraction between raw converted data and offset</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Raw converted Data, left aligned</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:0]</td>
<td>OFFSET[11:0]</td>
<td>Signed 12-bit data</td>
</tr>
<tr>
<td>01: 10-bit</td>
<td>DATA[11:2],00</td>
<td>OFFSET[11:0]</td>
<td>Signed 10-bit data</td>
</tr>
</tbody>
</table>
When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y = 1, 2, 3, 4) corresponding to the channel "i":

- If one of the offsets is enabled (bit OFFSETy_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

*Figure 67, Figure 68, Figure 69 and Figure 70* show alignments for signed and unsigned data.

### Table 81. Offset computation versus data resolution (continued)

<table>
<thead>
<tr>
<th>Resolution (bits RES[1:0])</th>
<th>Subtraction between raw converted data and offset</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10: 8-bit DATA[11:0],0000 OFFSET[11:0]</td>
<td>Signed 8-bit data</td>
<td>The user must configure OFFSET[3:0] to 0000</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 67.** Right alignment (offset disabled, unsigned value)

12-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

10-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

8-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

6-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
**Figure 68. Right alignment (offset enabled, signed value)**

12-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
</tr>
<tr>
<td>D11</td>
<td>D10</td>
<td>D9</td>
</tr>
<tr>
<td>D8</td>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
<td>D3</td>
</tr>
<tr>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

10-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
</tr>
<tr>
<td>D9</td>
<td>D8</td>
<td>D7</td>
</tr>
<tr>
<td>D6</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>D0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

8-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
</tr>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
</tr>
<tr>
<td>D4</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

6-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
<td>D3</td>
</tr>
<tr>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 69. Left alignment (offset disabled, unsigned value)**

12-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D11</td>
<td>D10</td>
<td>D9</td>
</tr>
<tr>
<td>D8</td>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
<td>D3</td>
</tr>
<tr>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

10-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D9</td>
<td>D8</td>
<td>D7</td>
</tr>
<tr>
<td>D6</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>D0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

8-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
</tr>
<tr>
<td>D4</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

6-bit data

<table>
<thead>
<tr>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>D0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
ADC overrun (OVR, OVRMOD)

The overrun flag (OSR) notifies of that a buffer overrun event occurred when the regular converted data has not been read (by the CPU or the DMA) before new converted data became available.

The OVR flag is set if the EOC flag is still 1 at the time when a new conversion completes. An interrupt can be generated if bit OVRIE = 1.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting bit ADSTP = 1.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overrun: the old data is maintained and the new conversion is discarded and lost. If OVR remains at 1, any further conversions occur but the result data is also 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, any further conversions operate normally and the ADC_DR register always contains the latest converted data.
Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD should be configured to 0 to manage overrun events as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set in the ADC_CFGR register, a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.
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: ADC overrun (OVR, OVRMOD)).

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 of the ADC_CFGR register:

- **DMA one shot mode (DMACFG = 0)**
  - This mode is suitable when the DMA is programmed to transfer a fixed number of data.
  - The mode is suitable when programming the DMA in circular mode.

- **DMA circular mode (DMACFG = 1)**
  - This mode is suitable when programming the DMA in circular mode.

### DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section) 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 with 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.
- 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 is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

### 16.4.27 Dynamic low-power features

#### Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY = 1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see Figure 72).
- For an injected conversion: when the JEOS bit has been cleared (see Figure 73).
This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

**Note:** There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

**Note:** This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see Figure 73).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see Figure 75).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see Figure 76).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS.
3. Set ADSTP.
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, CHANNELS = 1, 2, 3.
3. Injected configuration DISABLED.

Figure 72. AUTODLY = 1, regular conversion in continuous mode, software trigger

1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, CHANNELS = 1, 2, 3.
3. Injected configuration DISABLED.

Figure 73. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)
Figure 74. AUTDLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)

<table>
<thead>
<tr>
<th>Regular trigger</th>
<th>Ignored</th>
<th>Not ignored (occurs during injected sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC state</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_DR read access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_JDR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_JDR2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

by SW  by HW  Indicative timings

1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 01 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 1, CHANNELS = 5, 6
**Figure 75. AUTODLY = 1, regular continuous conversions interrupted by injected conversions**

1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN[1:0] = 01 (HW Trigger), JDISCEN = 0, CHANNELS = 5, 6

**Figure 76. AUTODLY = 1 in auto-injected mode (JAUTO = 1)**

1. AUTDLY = 1
2. Regular configuration: EXTEN[1:0] = 00 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6
16.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

![Analog watchdog guarded area](image)

**AWDx flag and interrupt**

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

**Description of analog watchdog 1**

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

*Table 82* shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

<table>
<thead>
<tr>
<th>Channels guarded by the analog watchdog</th>
<th>AWD1SGL bit</th>
<th>AWD1EN bit</th>
<th>JAWD1EN bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>All injected channels</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>All regular channels</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>All regular and injected channels</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Single(1) injected channel</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Single(1) regular channel</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Single(1) regular or injected channel</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. Selected by the AWD1CH(4:0) bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

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 bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[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 83* describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

### Table 83. Analog watchdog 1 comparison

<table>
<thead>
<tr>
<th>Resolution (bit RES[1:0])</th>
<th>Analog watchdog comparison between:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Raw converted data, left aligned</td>
<td>Thresholds</td>
</tr>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:0]</td>
<td>LT1[11:0] and HT1[11:0]</td>
</tr>
<tr>
<td>01: 10-bit</td>
<td>DATA[11:0],00</td>
<td>LT1[11:0] and HT1[11:0]</td>
</tr>
</tbody>
</table>

**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 corresponding bits in AWDxCH[18:0] (x=2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[18:0] (x=2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 84* describes how the comparison is performed for all the possible resolutions.

### Table 84. Analog watchdog 2 and 3 comparison

<table>
<thead>
<tr>
<th>Resolution (bits RES[1:0])</th>
<th>Analog watchdog comparison between:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Raw converted data, left aligned</td>
<td>Thresholds</td>
</tr>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:0]</td>
<td>LTx[7:0] and HTx[7:0]</td>
</tr>
<tr>
<td>01: 10-bit</td>
<td>DATA[11:0]</td>
<td>LTx[7:0] and HTx[7:0]</td>
</tr>
<tr>
<td>10: 8-bit</td>
<td>DATA[11:0]</td>
<td>LTx[7:0] and HTx[7:0]</td>
</tr>
<tr>
<td>11: 6-bit</td>
<td>DATA[11:0],00</td>
<td>LTx[7:0] and HTx[7:0]</td>
</tr>
</tbody>
</table>
**ADCy_AWDx_OUT signal output generation**

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y=ADC number, x=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_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).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS = 1). Note that stopping regular or injected conversions (setting ADSTP = 1 or JADSTP = 1) has no influence on the generation of ADCy_AWDx_OUT.

**Note:** ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag.

**Figure 78. ADCy_AWDx_OUT signal generation (on all regular channels)**

- Converting regular channels 1,2,3,4,5,6,7
- Regular channels 1,2,3,4,5,6,7 are all guarded
**Figure 79. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)**

- Converting regular channels 1,2,3,4,5,6,7
- Regular channels 1,2,3,4,5,6,7 are all guarded

**Figure 80. ADCy_AWDx_OUT signal generation (on a single regular channel)**

- Converting regular channels 1 and 2
- Only channel 1 is guarded

**Figure 81. ADCy_AWDx_OUT signal generation (on all injected channels)**

- Converting the injected channels 1, 2, 3, 4
- All injected channels 1, 2, 3, 4 are guarded
16.4.29 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to 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:

\[
\text{Result} = \frac{1}{M} \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)
\]

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_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to 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 intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.

**Figure 82. 20-bit to 16-bit result truncation**

**Figure 83** gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 83. Numerical example with 5-bit shift and rounding**

**Table 85** gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.
There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversions, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- 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 (EOS) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

### ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- 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 (AUTDLY)
- 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.

*Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).*

---

**Table 85. Maximum output results versus N and M (gray cells indicate truncation)**

<table>
<thead>
<tr>
<th>Over sampling 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 = 0011</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>0x1FFE</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
<td>0x0020</td>
<td>0x0010</td>
</tr>
<tr>
<td>4x</td>
<td>0x3FFC</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
<td>0x0020</td>
</tr>
<tr>
<td>8x</td>
<td>0x7FF8</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
</tr>
<tr>
<td>16x</td>
<td>0xFFFF0</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
</tr>
<tr>
<td>32x</td>
<td>0x1FFE0</td>
<td>0xFFE0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
</tr>
<tr>
<td>64x</td>
<td>0x3FFC0</td>
<td>0xFFC0</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
<tr>
<td>128x</td>
<td>0x7FF80</td>
<td>0xFFFF8</td>
<td>0x3FFC0</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
</tr>
<tr>
<td>256x</td>
<td>0xFFFF00</td>
<td>0xFFFF0</td>
<td>0x7FF80</td>
<td>0x3FFC0</td>
<td>0xFFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0800</td>
<td>0x0400</td>
</tr>
</tbody>
</table>
Analog watchdog

The analog watchdog functionality is maintained, with the following difference:

- The RES[1:0] bits are ignored, comparison is always done using the full 12-bit values HT[11:0] and LT[11:0]
- The comparison is performed on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4]

Note: Care must be taken when using high shifting values, since 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 HT[0:7] / LT[0:7], and HT[11:8] / LT[11:8] must be kept reset.

Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful 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 TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 84 below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 84. Triggered regular oversampling mode (TROVS bit = 1)

Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).
**Oversampling regular channels only**

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);

- In resumed mode, the accumulation restarts from 0 (previous conversion results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequencer is blocked.

*Figure 85* gives examples for a 4x oversampling ratio.

**Figure 85. Regular oversampling modes (4x ratio)**

![Diagram showing regular oversampling modes](image)

---

**Oversampling Injected channels only**

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.
Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on Figure 86 below.

**Figure 86. Regular and injected oversampling modes used simultaneously**

<table>
<thead>
<tr>
<th>Regular channels</th>
<th>Injected channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ch(N)0</td>
<td>Ch(N)1</td>
</tr>
<tr>
<td>Trigger</td>
<td>Trigger</td>
</tr>
<tr>
<td>oversampling resumed</td>
<td></td>
</tr>
</tbody>
</table>

Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on Figure 87 below.

**Figure 87. Triggered regular oversampling with injection**

<table>
<thead>
<tr>
<th>Regular channels</th>
<th>Injected channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ch(N)0</td>
<td>Ch(N)1</td>
</tr>
<tr>
<td>Trigger</td>
<td>Trigger</td>
</tr>
<tr>
<td>oversampling resumed</td>
<td></td>
</tr>
</tbody>
</table>

| ROVSE = 1, JOVSE = 0, ROVSM = 1, TROVS = 1 |
Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The Figure 88 below shows how the conversions are sequenced.

**Figure 88. Oversampling in auto-injected mode**

<table>
<thead>
<tr>
<th>Regular channels</th>
<th>Injected channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>N0, N1, N2, N3</td>
<td>J0, J1, J2, J3</td>
</tr>
<tr>
<td>K0, K1, K2, K3</td>
<td>L0, L1, L2, L3</td>
</tr>
</tbody>
</table>

JAUTO = 1, ROVSE = 1, JOVSE = 1, ROVSM = X, TROVS = 0

It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

16.4.30 Temperature sensor

The temperature sensor can be used to measure the junction temperature (Tj) of the device. The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value. When not in use, the sensor can be put in power down mode. It support the temperature range –40 to 125 °C.

**Figure 89** shows the block diagram of connections between the temperature sensor and the ADC.

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).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor’s output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

**Figure 89** shows the block diagram of the temperature sensor.
Reading the temperature

To use the sensor:
1. Select the ADC input channels that is connected to $V_{TS}$.
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the CH17SEL bit in the ADCx_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting $V_{TS}$ data in the ADC data register.
6. Calculate the actual temperature using the following formula:

\[
\text{Temperature in } ^\circ\text{C} = \frac{\text{TS\_CAL2\_TEMP} - \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times \frac{\text{TS\_DATA} - \text{TS\_CAL1} + \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}}
\]

Where:
- $\text{TS\_CAL2}$ is the temperature sensor calibration value acquired at $\text{TS\_CAL2\_TEMP}$.
- $\text{TS\_CAL1}$ is the temperature sensor calibration value acquired at $\text{TS\_CAL1\_TEMP}$.
- $\text{TS\_DATA}$ is the actual temperature sensor output value converted by ADC.
- Refer to the device datasheet for more information about $\text{TS\_CAL1}$ and $\text{TS\_CAL2}$ 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 CH17SEL bits should be set at the same time.

The above formula is given for $\text{TS\_DATA}$ measurement done with the same $V_{\text{REF+}}$ voltage as $\text{TS\_CAL1}/\text{TS\_CAL2}$ values. If $V_{\text{REF+}}$ is different, the formula must be adapted. For example if $V_{\text{REF+}} = 3.3$ V and $\text{TS\_CAL}$ data are acquired at $V_{\text{REF+}} = 3.0$ V, $\text{TS\_DATA}$ must be replaced by $\text{TS\_DATA} \times (3.3/3.0)$. 
16.4.31 **V\textsubscript{BAT} supply monitoring**

The CH18SEL bit in the ADC\textsubscript{x}\_CCR register is used to switch to the battery voltage. As the \textsubscript{V}\textsubscript{BAT} voltage could be higher than \textsubscript{V}\textsubscript{DDA}, to ensure the correct operation of the ADC, the \textsubscript{V}\textsubscript{BAT} pin is internally connected to a bridge divider by 3. This bridge is automatically enabled when CH18SEL is set, to connect \textsubscript{V}\textsubscript{BAT}/3 to the ADC input channels. As a consequence, the converted digital value is one third of the \textsubscript{V}\textsubscript{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the \textsubscript{V}\textsubscript{BAT}/3 voltage.

The figure below shows the block diagram of the \textsubscript{V}\textsubscript{BAT} sensing feature.

![Figure 90. \textsubscript{V}\textsubscript{BAT} channel block diagram](image)

1. The CH18SEL bit must be set to enable the conversion of internal channel for \textsubscript{V}\textsubscript{BAT}/3.

16.4.32 **Monitoring the internal voltage reference**

It is possible to monitor the internal voltage reference (\textsubscript{V}\textsubscript{REF\textsubscript{INT}}) to have a reference point for evaluating the ADC \textsubscript{V}\textsubscript{REF+} voltage level.

The internal voltage reference is internally connected to the input channel 0 of the ADC1 (ADC\textsubscript{1\_INP0}).

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

*Figure 91* shows the block diagram of the \textsubscript{V}\textsubscript{REF\textsubscript{INT}} sensing feature.
1. The VREFEN bit into ADC_CCR register must be set to enable the conversion of internal channels (\(V_{\text{REFINT}}\)).

**Calculating the actual \(V_{\text{REF+}}\) voltage using the internal reference voltage**

The power supply voltage applied to the device may be subject to variations or not precisely known. When \(V_{\text{DDA}}\) is connected to \(V_{\text{REF+}}\), it is possible to compute the actual \(V_{\text{DDA}}\) voltage using the embedded internal reference voltage \(V_{\text{REFINT}}\). \(V_{\text{REFINT}}\) and its calibration data, acquired by the ADC during the manufacturing process at \(V_{\text{DDA,Charac}}\), can be used to evaluate the actual \(V_{\text{DDA}}\) voltage level.

The following formula gives the actual \(V_{\text{REF+}}\) voltage supplying the device:

\[
V_{\text{REF+}} = V_{\text{REF+ Charac}} \cdot V_{\text{REFINT CAL}} / V_{\text{REFINT DATA}}
\]

Where:

- \(V_{\text{REF+ Charac}}\) is the value of \(V_{\text{REF+}}\) voltage characterized at \(V_{\text{REFINT}}\) during the manufacturing process. It is specified in the device datasheet.
- \(V_{\text{REFINT CAL}}\) is the \(V_{\text{REFINT}}\) calibration value
- \(V_{\text{REFINT DATA}}\) is the actual \(V_{\text{REFINT}}\) 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 \(V_{\text{REF+}}\) and the voltage applied on the converted channel.

For most applications \(V_{\text{DDA}}\) value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from \(V_{\text{DDA}}\):

\[
V_{\text{CHANNELx}} = \frac{V_{\text{REF+}}}{\text{FULL SCALE}} \cdot \text{ADC DATA}
\]
By replacing $V_{REF+}$ by the formula provided above, the absolute voltage value is given by the following formula:

$$V_{\text{CHANNELx}} = \frac{V_{\text{REF+ Charac}} \cdot V_{\text{REFINT_CAL}} \cdot \text{ADC_DATA}}{V_{\text{REFINT_DATA}} \cdot \text{FULL_SCALE}}$$

For applications where $V_{REF+}$ is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{\text{CHANNELx}} = \frac{V_{\text{REF+}}}{\text{FULL_SCALE}} \cdot \text{ADC_DATA}$$

Where:
- $V_{\text{REF+ Charac}}$ is the value of $V_{REF+}$ voltage characterized at $V_{\text{REFINT}}$ during the manufacturing process.
- $V_{\text{REFINT_CAL}}$ is the $V_{\text{REFINT}}$ calibration value
- ADC_DATA is the value measured by the ADC on channel x (right-aligned)
- $V_{\text{REFINT_DATA}}$ is the actual $V_{\text{REFINT}}$ output value converted by the ADC
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it is $2^{12} - 1 = 4095$ or with 8-bit resolution, $2^8 - 1 = 255$.

**Note:** If ADC measurements are done using an output format other than 16-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

### 16.5 ADC in low-power mode

**Table 86. Effect of low-power modes on the ADC**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. DMA requests are functional.</td>
</tr>
<tr>
<td>Low-power run</td>
<td>No effect. DMA requests are functional.</td>
</tr>
<tr>
<td>Low-power sleep</td>
<td>No effect. DMA requests are functional.</td>
</tr>
<tr>
<td>Stop 0/Stop 1</td>
<td>The ADC is not operational. Its state is kept. The ADC consumes the static current recommended to disable the peripheral in advance in order to reduce power consumption.</td>
</tr>
<tr>
<td>Stop 2</td>
<td>The ADC is not operational. Its state is kept. The ADC regulator is disabled by hardware. ADC and regulator must be disabled before entering Stop 2 mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The ADC is powered down and must be reinitialized after exiting Standby or Shutdown mode.</td>
</tr>
</tbody>
</table>
16.6 ADC interrupts

An interrupt can be generated:
- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 87. ADC interrupts

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC ready</td>
<td>ADRDY</td>
<td>ADRDYIE</td>
</tr>
<tr>
<td>End of conversion of a regular group</td>
<td>EOC</td>
<td>EOCIE</td>
</tr>
<tr>
<td>End of sequence of conversions of a regular group</td>
<td>EOS</td>
<td>EOSIE</td>
</tr>
<tr>
<td>End of conversion of a injected group</td>
<td>JEOC</td>
<td>JEOSIE</td>
</tr>
<tr>
<td>End of sequence of conversions of an injected group</td>
<td>JEOS</td>
<td>JEOSIE</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>End of sampling phase</td>
<td>EOSMP</td>
<td>EOSMPIE</td>
</tr>
<tr>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
</tr>
<tr>
<td>Injected context queue overflows</td>
<td>JQOVF</td>
<td>JQOVFIE</td>
</tr>
</tbody>
</table>
16.7 ADC registers

Refer to Section 1.2 on page 61 for a list of abbreviations used in register descriptions.

16.7.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00
Reset value: 0x0000 0000

| Bit 31:11 | Reserved, must be kept at reset value. |
| Bit 10  | JQOVF: Injected context queue overflow |
| This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to Section 16.4.21: Queue of context for injected conversions for more information. |
| 0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software) |
| 1: Injected context queue overflow has occurred |
| Bit 9   | AWD3: Analog watchdog 3 flag |
| This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it. |
| 0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software) |
| 1: Analog watchdog 3 event occurred |
| Bit 8   | AWD2: Analog watchdog 2 flag |
| This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it. |
| 0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software) |
| 1: Analog watchdog 2 event occurred |
| Bit 7   | AWD1: Analog watchdog 1 flag |
| This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it. |
| 0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software) |
| 1: Analog watchdog 1 event occurred |
| Bit 6   | JEOS: Injected channel end of sequence flag |
| This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it. |
| 0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software) |
| 1: Injected conversions complete |
Bit 5  **JEOC**: Injected channel end of conversion flag
This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register.
0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)
1: Injected channel conversion complete

Bit 4  **OVR**: ADC overrun
This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed 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 regular sequence flag
This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.
0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)
1: Regular Conversions sequence complete

Bit 2  **EOC**: End of conversion flag
This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.
0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)
1: Regular channel conversion complete

Bit 1  **EOSMP**: End of sampling flag
This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.
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
16.7.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04
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>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 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled
1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled
1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which 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 2 interrupt.

0: Analog watchdog 2 interrupt disabled
1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which 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 1 interrupt.

0: Analog watchdog 1 interrupt disabled
1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled
1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).
Bit 5 **JEOCIE**: End of injected conversion interrupt enable
   This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.
   0: JEOC interrupt disabled.
   1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.
   Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable
   This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.
   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 = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable
   This bit is set and cleared by software to enable/disable the end of regular 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 = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable
   This bit is set and cleared by software to enable/disable the end of a regular 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 = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions
   This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.
   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 = 0 (which ensures that no regular 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 = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
16.7.3 ADC control register (ADC_CR)

Address offset: 0x08
Reset value: 0x2000 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>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 ADCAL: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode. It is cleared by hardware after calibration is complete.

0: Calibration complete
1: Write 1 to calibrate the ADC. Read 1 means that a calibration is in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0. The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).

Bit 30 ADCALDIF: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL launches a calibration in single-ended inputs mode.
1: Writing ADCAL launches a calibration in differential inputs mode.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 DEEPPWD: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down
1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 28 ADVREGEN: ADC voltage regulator enable

This bit is set by software to enable the ADC voltage regulator.
Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled
1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to Section 16.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN).

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).
Bits 27:6  Reserved, must be kept at reset value.

Bit 5  **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing
1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

**Note:** The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 4  **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing
1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

**Note:** The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 3  **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (JEXTSEL = 0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.
– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.
1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

**Note:** The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared).
Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:
- in single conversion mode when software trigger is selected (EXTSEL = 0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.
1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

*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).

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

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*: The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY 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, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)
16.7.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C
Reset value: 0x8000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>JQDIS: Injected Queue disable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits are set and cleared by software to disable the Injected Queue mechanism:</td>
</tr>
<tr>
<td>0</td>
<td>Injected Queue enabled</td>
</tr>
<tr>
<td>1</td>
<td>Injected Queue disabled</td>
</tr>
<tr>
<td>Note:</td>
<td>The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing). A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30:26</th>
<th>AWD1CH[4:0]: Analog watchdog 1 channel selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.</td>
<td></td>
</tr>
<tr>
<td>00000: ADC analog input channel 0 monitored by AWD1</td>
<td></td>
</tr>
<tr>
<td>00001: ADC analog input channel 1 monitored by AWD1</td>
<td></td>
</tr>
<tr>
<td>.....</td>
<td></td>
</tr>
<tr>
<td>10010: ADC analog input channel 18 monitored by AWD1</td>
<td></td>
</tr>
<tr>
<td>others: reserved, must not be used</td>
<td></td>
</tr>
<tr>
<td>Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.</td>
<td></td>
</tr>
<tr>
<td>The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.</td>
<td></td>
</tr>
<tr>
<td>The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>JAUTO: Automatic injected group conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Automatic injected group conversion disabled</td>
</tr>
<tr>
<td>1</td>
<td>Automatic injected group conversion enabled</td>
</tr>
<tr>
<td>Note:</td>
<td>The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 24</th>
<th>JAWD1EN: Analog watchdog 1 enable on injected channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is set and cleared by software</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Analog watchdog 1 disabled on injected channels</td>
</tr>
<tr>
<td>1</td>
<td>Analog watchdog 1 enabled on injected channels</td>
</tr>
<tr>
<td>Note:</td>
<td>The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).</td>
</tr>
</tbody>
</table>
Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels
This bit is set and cleared by software
0: Analog watchdog 1 disabled on regular channels
1: Analog watchdog 1 enabled on regular channels
*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 22 **AWD1SGL**: Enable the watchdog 1 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 AWD1CH[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 these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 21 **JQM**: JSQR queue mode
This bit is set and cleared by software.
It defines how an empty Queue is managed.
0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.
1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.
Refer to Section 16.4.21: Queue of context for injected conversions for more information.
*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 20 **JDISCEN**: Discontinuous mode on injected channels
This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.
0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled
*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*
*It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.*

Bits 19:17 **DISCNUM[2:0]**: Discontinuous mode channel count
These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.
000: 1 channel
001: 2 channels
...
111: 8 channels
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*
**Bit 16 DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode for regular channels.

- 0: Discontinuous mode for regular channels disabled
- 1: Discontinuous mode for regular channels enabled

**Note**: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

**Bit 15** Reserved, must be kept at reset value.

**Bit 14 AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

- 0: Auto-delayed conversion mode off
- 1: Auto-delayed conversion mode on

**Note**: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

**Bit 13 CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular 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 DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

**Bit 12 OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is 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 = 0 (which ensures that no regular conversion is ongoing).

**Bits 11:10 EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

- 00: Hardware trigger detection disabled (conversions can be launched 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 these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).
Bits 9:6 **EXTSEL[3:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

- 0000: Event 0
- 0001: Event 1
- 0010: Event 2
- 0011: Event 3
- 0100: Event 4
- 0101: Event 5
- 0110: Event 6
- 0111: Event 7
- ... 1111: Event 15

**Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to Section: Data register, data alignment and offset (ADC_DR, OFFSETy, OFFSETy_CH, ALIGN)

- 0: Right alignment
- 1: Left alignment

**Note:** The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which 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-bit
- 01: 10-bit
- 10: 8-bit
- 11: 6-bit

**Note:** The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 2 Reserved, must be kept at reset value.

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: Managing conversions using the DMA

**Note:** The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which 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 to manage automatically the converted data. For more details, refer to Section: Managing conversions using the DMA.

- 0: DMA disabled
- 1: DMA enabled

**Note:** The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
16.7.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:28</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>27:17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>16:11</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>ROVSM: Regular Oversampling mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software to select the regular oversampling mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> 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>9</td>
<td>TROVS: Triggered Regular Oversampling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software to enable triggered oversampling</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: All oversampled conversions for a channel are done consecutively following a trigger</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Each oversampled conversion for a channel needs a new trigger</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> 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>8:5</td>
<td>OVSS[3:0]: Oversampling shift</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0000: No shift</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0001: Shift 1-bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0010: Shift 2-bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0011: Shift 3-bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0100: Shift 4-bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0101: Shift 5-bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0110: Shift 6-bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0111: Shift 7-bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000: Shift 8-bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other codes reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).</td>
<td></td>
</tr>
</tbody>
</table>
Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the 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 these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 1 **JOVSE**: Injected Oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

- 0: Injected Oversampling disabled
- 1: Injected Oversampling enabled

**Note:** The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular Oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

- 0: Regular Oversampling disabled
- 1: Regular Oversampling enabled

**Note:** The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

### 16.7.6 ADC sample time register 1 (ADC_SMPR1)

**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>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>
</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).
Bits 31:30  Reserved, must be kept at reset value.

Bits 29:0  **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles  
001: 6.5 ADC clock cycles  
010: 12.5 ADC clock cycles  
011: 24.5 ADC clock cycles  
100: 47.5 ADC clock cycles  
101: 92.5 ADC clock cycles  
110: 247.5 ADC clock cycles  
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing). Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

**16.7.7  ADC sample time register 2 (ADC_SMPR2)**

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></td>
<td></td>
<td></td>
<td></td>
<td></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 31:27  Reserved, must be kept at reset value.

Bits 26:0  **SMPx[2:0]:** Channel x sampling time selection (x = 18 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles  
001: 6.5 ADC clock cycles  
010: 12.5 ADC clock cycles  
011: 24.5 ADC clock cycles  
100: 47.5 ADC clock cycles  
101: 92.5 ADC clock cycles  
110: 247.5 ADC clock cycles  
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing). Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.
### 16.7.8 ADC watchdog threshold register 1 (ADC_TR1)

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>

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 1.  
Refer to Section 16.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)  
**Note:** The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

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 1.  
Refer to Section 16.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)  
**Note:** The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

### 16.7.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24  
Reset value: 0x00FF 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 31:24  Reserved, must be kept at reset value.

Bits 23:16  **HT2[7:0]**: Analog watchdog 2 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 2.
Refer to *Section 16.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **LT2[7:0]**: Analog watchdog 2 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog 2.
Refer to *Section 16.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 16.7.10 ADC watchdog threshold register 3 (ADC_TR3)

**Address offset:** 0x28  
**Reset value:** 0x00FF 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></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>
<tr>
<td>LT3[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>
<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></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 23:16  **HT3[7:0]**: Analog watchdog 3 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 3.
Refer to *Section 16.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **LT3[7:0]**: Analog watchdog 3 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog 3.
This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*
### 16.7.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:29</th>
<th>Bit 28:24</th>
<th>Bit 23</th>
<th>Bit 22:18</th>
<th>Bit 17</th>
<th>Bit 16:12</th>
<th>Bit 11</th>
<th>Bit 10:6</th>
<th>Bit 5:4</th>
<th>Bit 3:0</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>
</tr>
</tbody>
</table>

Bits 31:29  Reserved, must be kept at reset value.

Bits 28:24  **SQ4[4:0]: 4th conversion in regular sequence**  
These bits are written by software with the channel number (0 to 18) assigned as the 4th in the regular conversion sequence.  
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 23  Reserved, must be kept at reset value.

Bits 22:18  **SQ3[4:0]: 3rd conversion in regular sequence**  
These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the regular conversion sequence.  
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 17  Reserved, must be kept at reset value.

Bits 16:12  **SQ2[4:0]: 2nd conversion in regular sequence**  
These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the regular conversion sequence.  
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 11  Reserved, must be kept at reset value.

Bits 10:6  **SQ1[4:0]: 1st conversion in regular sequence**  
These bits are written by software with the channel number (0 to 18) assigned as the 1st in the regular conversion sequence.  
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bits 5:4  Reserved, must be kept at reset value.

Bits 3:0  **L[3:0]: Regular channel sequence length**  
These bits are written by software to define the total number of conversions in the regular channel conversion sequence.  
0000: 1 conversion  
0001: 2 conversions  
...  
1111: 16 conversions  
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

*Note: Some channels are not connected physically and must not be selected for conversion.*
16.7.12 ADC regular sequence register 2 (ADC_SQR2)

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]**: 9th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 9th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]**: 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 8th in the regular conversion sequence

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]**: 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 7th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]**: 6th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 6th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]**: 5th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 5th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

*Note:* Some channels are not connected physically and must not be selected for conversion.
16.7.13  ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38
Reset value: 0x0000 0000

| Bits 31:29  | Reserved, must be kept at reset value. |
| Bits 28:24  | SQ14[4:0]: 14th conversion in regular sequence |
|            | These bits are written by software with the channel number (0 to 18) assigned as the 14th in the regular conversion sequence. |
|            | **Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing). |
| Bit 23     | Reserved, must be kept at reset value. |
| Bits 22:18 | SQ13[4:0]: 13th conversion in regular sequence |
|            | These bits are written by software with the channel number (0 to 18) assigned as the 13th in the regular conversion sequence. |
|            | **Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing). |
| Bit 17     | Reserved, must be kept at reset value. |
| Bits 16:12 | SQ12[4:0]: 12th conversion in regular sequence |
|            | These bits are written by software with the channel number (0 to 18) assigned as the 12th in the regular conversion sequence. |
|            | **Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing). |
| Bit 11     | Reserved, must be kept at reset value. |
| Bits 10:6  | SQ11[4:0]: 11th conversion in regular sequence |
|            | These bits are written by software with the channel number (0 to 18) assigned as the 11th in the regular conversion sequence. |
|            | **Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing). |
| Bit 5      | Reserved, must be kept at reset value. |
| Bits 4:0   | SQ10[4:0]: 10th conversion in regular sequence |
|            | These bits are written by software with the channel number (0 to 18) assigned as the 10th in the regular conversion sequence. |
|            | **Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing). |

**Note:** Some channels are not connected physically and must not be selected for conversion.
### 16.7.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C  
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>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 31:11  Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]**: 16th conversion in regular sequence

- These bits are written by software with the channel number (0 to 18) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5  Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]**: 15th conversion in regular sequence

- These bits are written by software with the channel number (0 to 18) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

**Note:** Some channels are not connected physically and must not be selected for conversion.

### 16.7.15 ADC regular data register (ADC_DR)

Address offset: 0x40  
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>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>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
<td>Rd</td>
</tr>
</tbody>
</table>

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

Bits 15:0 **RDATA[15:0]**: Regular data converted

- These bits are read-only. They contain the conversion result from the last converted regular channel.
- The data are left- or right-aligned as described in Section 16.4.26: Data management.
16.7.16 ADC injected sequence register (ADC_JSQR)

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>Res.</td>
<td>JSQ4[4:0]</td>
<td>Res.</td>
<td>JSQ3[4:0]</td>
<td>Res.</td>
<td>JSQ2[4:2]</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>Res.</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>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>

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **JSQ4[4:0]**: 4th conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 4th in the injected conversion sequence.

*Note:* The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 25 Reserved, must be kept at reset value.

Bits 24:20 **JSQ3[4:0]**: 3rd conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the injected conversion sequence.

*Note:* The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 19 Reserved, must be kept at reset value.

Bits 18:14 **JSQ2[4:0]**: 2nd conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the injected conversion sequence.

*Note:* The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 13 Reserved, must be kept at reset value.

Bits 12:8 **JSQ1[4:0]**: 1st conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 1st in the injected conversion sequence.

*Note:* The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).
Bits 7:6 JEXTEN[1:0]: External Trigger Enable and Polarity Selection for injected channels
These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.
- 00: If JQDIS = 0 (queue enabled), Hardware and software trigger detection disabled
- 00: If JQDIS = 1 (queue disabled), Hardware trigger detection disabled (conversions can be launched 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 these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 16.4.21: Queue of context for injected conversions)

Bits 5:2 JEXTSEL[3:0]: External Trigger Selection for injected group
These bits select the external event used to trigger the start of conversion of an injected group:
- 0000: Event 0
- 0001: Event 1
- 0010: Event 2
- 0011: Event 3
- 0100: Event 4
- 0101: Event 5
- 0110: Event 6
- 0111: Event 7
- ... 1111: Event 15

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 JL[1:0]: Injected channel sequence length
These bits are written by software to define the total number of conversions in the injected channel conversion sequence.
- 00: 1 conversion
- 01: 2 conversions
- 10: 3 conversions
- 11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.
16.7.17 ADC offset y register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y - 1), (y= 1 to 4)
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></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>

Bit 31 **OFFSET_EN**: Offset y enable
This bit is written by software to enable or disable the offset programmed into bits OFFSETy[11:0].

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSET_CH[4:0]**: Channel selection for the data offset y
These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[11:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the data offset y.

Bits 25:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET[11:0]**: Data offset y for the channel programmed into bits OFFSETy_CH[4:0]
These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.

Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[11:0] which is subtracted when converting channel 4.
16.7.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)
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>

JDATA[15:0] = 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 15:0 JDATA[15:0]: Injected data
These bits are read-only. They contain the conversion result from injected channel y. The data are left-or right-aligned as described in Section 16.4.26: Data management.

16.7.19 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>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>

AWD2CH[15:0] = rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

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

Bits 18:0 AWD2CH[18:0]: Analog watchdog 2 channel selection
These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.
AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2
AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2
When AWD2CH[18:0] = 000...0, the analog watchdog 2 is disabled

Note: The channels selected by AWD2CH must also be selected into the SQRi or JSQRi registers.
The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
Some channels are not connected physically and must not be selected for the analog watchdog.
16.7.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset: 0xA4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
</tbody>
</table>

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

### Bits 18:0 **AWD3CH[18:0]**: Analog watchdog 3 channel selection
- These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.
- **AWD3CH[i]** = 0: ADC analog input channel i is not monitored by AWD3
- **AWD3CH[i]** = 1: ADC analog input channel i is monitored by AWD3
- When AWD3CH[18:0] = 000..0, the analog watchdog 3 is disabled

**Note:** The channels selected by AWD3CH must be also selected into the SQRi or JSQRi registers.
- The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
- Some channels are not connected physically and must not be selected for the analog watchdog.

16.7.21 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0xB0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset: 0xB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value: 0x0000 0000</td>
</tr>
</tbody>
</table>

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

### Bits 18:0 **DIFSEL[18:0]**: Differential mode for channels 18 to 0.
- These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential mode.
- **DIFSEL[i]** = 0: ADC analog input channel i is configured in single ended mode
- **DIFSEL[i]** = 1: ADC analog input channel i is configured in differential mode

**Note:** The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).
- The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).
16.7.22 ADC calibration factors (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>
<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.

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

- These bits are written by hardware or by software.
- Once a differential inputs 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 is then applied once a new differential calibration is launched.

*Note:* The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In single-ended mode

- These bits are written by hardware or by software.
- Once a single-ended inputs 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 is then applied once a new single-ended calibration is launched.

*Note:* The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

16.8 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

16.8.1 ADC common status register (ADC_CSR)

Address offset: 0x300
Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.
16.8.2 ADC common control 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>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CH1SEL</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>VREF</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>PRESC[3:0]</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>CKMODE[1:0]</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</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>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 JQOVF_MST: Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 AWD3_MST: Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 AWD2_MST: Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 AWD1_MST: Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 JEOS_MST: End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 JEOC_MST: End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 OVR_MST: Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 EOS_MST: End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 EOC_MST: End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 EOSMP_MST: End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 ADRDY_MST: Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.
Bits 31:25  Reserved, must be kept at reset value.

Bit 24  **CH18SEL**: CH18 selection
This bit is set and cleared by software to control channel 18.
0: \(V_{\text{BAT}}\) channel disabled.
1: \(V_{\text{BAT}}\) channel enabled

Bit 23  **CH17SEL**: CH17 selection
This bit is set and cleared by software to control channel 17.
0: Temperature sensor channel disabled
1: Temperature sensor channel enabled

Bit 22  **VREFEN**: \(V_{\text{REFINT}}\) enable
This bit is set and cleared by software to enable/disable the \(V_{\text{REFINT}}\) channel.
0: \(V_{\text{REFINT}}\) channel disabled
1: \(V_{\text{REFINT}}\) channel enabled

Bits 21:18  **PRESC[3:0]**: ADC prescaler
These bits are set and cleared by software to select the frequency of the clock to the ADC.
The clock is common for all the ADCs.
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:* The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 00.

Bits 17:16  **CKMODE[1:0]**: ADC clock mode
These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):
00: \(\text{CK}_{\text{ADCx}}\) (x = 123) (Asynchronous clock mode), generated at product level (refer to Section 6: Reset and clock control (RCC))
01: HCLK/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set (HPRE[3:0] = 0xxx in RCC_CFGR register) and if the system clock has a 50% duty cycle.
10: HCLK/2 (Synchronous clock mode)
11: HCLK/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note:* The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 15:0  Reserved, must be kept at reset value.
# 16.9 ADC register map

Table 88. ADC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Address (Hex)</th>
<th>Bits</th>
<th>Description</th>
<th>Value (reset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>ADC_ISR</td>
<td>0x40</td>
<td>7</td>
<td>Interrupt status register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>ADC_IER</td>
<td>0x44</td>
<td>7</td>
<td>Interrupt enable register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>ADC_CR</td>
<td>0x48</td>
<td>16</td>
<td>Control register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_CFGR</td>
<td>0x4C</td>
<td>16</td>
<td>Configuration register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>ADC_CFGR2</td>
<td>0x50</td>
<td>16</td>
<td>Configuration register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>ADC_SMPR1</td>
<td>0x54</td>
<td>16</td>
<td>Sample register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>ADC_SMPR2</td>
<td>0x58</td>
<td>16</td>
<td>Sample register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>ADC_TR1</td>
<td>0x5C</td>
<td>16</td>
<td>Transfer register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>ADC_TR2</td>
<td>0x60</td>
<td>16</td>
<td>Transfer register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>ADC_TR3</td>
<td>0x64</td>
<td>16</td>
<td>Transfer register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>ADC_SQR1</td>
<td>0x68</td>
<td>16</td>
<td>Square register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>ADC_SQR2</td>
<td>0x6C</td>
<td>16</td>
<td>Square register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x30</td>
<td>ADC_SQR3</td>
<td>0x70</td>
<td>16</td>
<td>Square register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x34</td>
<td>ADC_SQR4</td>
<td>0x74</td>
<td>16</td>
<td>Square register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x38</td>
<td>ADC_DR</td>
<td>0x78</td>
<td>8</td>
<td>Data register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x40</td>
<td>ADC_NR</td>
<td>0x7C</td>
<td>8</td>
<td>Result register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44</td>
<td>ADC_NR1</td>
<td>0x80</td>
<td>8</td>
<td>Result register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48</td>
<td>ADC_NR2</td>
<td>0x84</td>
<td>8</td>
<td>Result register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x50</td>
<td>ADC_NR3</td>
<td>0x88</td>
<td>8</td>
<td>Result register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x54</td>
<td>ADC_NR4</td>
<td>0x8C</td>
<td>8</td>
<td>Result register</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Note: The table shows the ADC register map and reset values with specific bit fields and reset values for each register.]
### Table 88. ADC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset1 EN</th>
<th>Offset2 EN</th>
<th>Offset3 EN</th>
<th>Offset4 EN</th>
<th>DIFSEL EN</th>
<th>AWD2CH</th>
<th>AWD3CH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reset value</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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x50- 0x5C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x60</td>
<td>ADC_OFR1</td>
<td>OFFSET1_EN</td>
<td>OFFSET2_EN</td>
<td>OFFSET3_EN</td>
<td>OFFSET4_EN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x64</td>
<td>ADC_OFR2</td>
<td>OFFSET1_EN</td>
<td>OFFSET2_EN</td>
<td>OFFSET3_EN</td>
<td>OFFSET4_EN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x68</td>
<td>ADC_OFR3</td>
<td>OFFSET1_EN</td>
<td>OFFSET2_EN</td>
<td>OFFSET3_EN</td>
<td>OFFSET4_EN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x6C</td>
<td>ADC_OFR4</td>
<td>OFFSET1_EN</td>
<td>OFFSET2_EN</td>
<td>OFFSET3_EN</td>
<td>OFFSET4_EN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x70- 0x7C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>ADC_JDR1</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 0 0 0 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>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x84</td>
<td>ADC_JDR2</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 0 0 0 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>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x88</td>
<td>ADC_JDR3</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 0 0 0 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>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8C</td>
<td>ADC_JDR4</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 0 0 0 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>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8C- 0x9C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA0</td>
<td>ADC_AWD2CR</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 0 0 0 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>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xA4</td>
<td>ADC_AWD3CR</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 0 0 0 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>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x8- 0x9C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
Table 88. ADC register map and reset values (continued)

| Offset | Register name 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  |
|--------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xB4   | ADC_CALFACT               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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  |

Table 89. ADC register map and reset values (master and slave ADC common registers) offset = 0x300

| Offset | Register name 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  |
|--------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | ADC_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  | 0  | 0  | 0  | 0  | 0  |
| 0x04   | Reserved                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| Offset | Register name 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  |
|--------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x08   | ADC_CCR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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 on page 66 for the register boundary addresses.
17 Voltage reference buffer (VREFBUF)

17.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADC 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).

17.2 VREFBUF implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREFBUF</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

17.3 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>VREFBUF buffer configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>VREFBUF buffer off mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– ( V_{REF} ) pin pulled-down to ( V_{SSA} )</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>– ( V_{REF} ) 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>– ( V_{REF} ) 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>– VREF is enable without output buffer, VREF+ pin voltage is hold 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>

\(^a\) The minimum \( V_{DDA} \) voltage depends on VRS setting, refer to the product datasheet.
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.

17.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. For the VRS = 1 setting, the calibration data is automatically loaded to the TRIM register at reset. For the VRS = 0 setting, the software must take care of copying the calibration data from the read-only system memory area (flash memory) to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits.

<table>
<thead>
<tr>
<th>Calibration value name</th>
<th>Description</th>
<th>Memory address</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREF_SC0</td>
<td>VREFBUF trimming value for VRS = 0</td>
<td>0x1FFF 75F0</td>
</tr>
<tr>
<td>VREF_SC1</td>
<td>VREFBUF trimming value for VRS = 1</td>
<td>0x1FFF 7530</td>
</tr>
</tbody>
</table>

17.5 VREFBUF registers

17.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00
Reset value: 0x0000 0002

<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>
</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 VREF_OUT1 (around 2.048 V).
1: Voltage reference set to VREF_OUT2 (around 2.5 V).

Bit 1 **HIZ**: High impedance mode
This bit controls the analog switch to connect or not the VREF+ pin.
0: VREF+ pin is internally connected to the voltage reference buffer output.
1: VREF+ pin is high impedance.
Refer to **Table 91: 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.

### 17.5.2 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>
</tbody>
</table>

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

Bits 5:0 **TRIM[5:0]**: Trimming code
For VRS = 1 (2.5 V), these bits are automatically initialized after reset with the trimming value stored in the flash memory during the production test.
For VRS = 0 (2.048V), the software must take care of copying the calibration data from the read-only system memory area (flash memory) to the TRIM[5:0] bitfield. Writing into these bits allows the tuning of the internal reference buffer voltage.

Note: If the user application performs the trimming, the trimming code should start from 000000 to 111111 in ascending order.

### 17.5.3 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>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>VREFBUF_CSR</td>
</tr>
</tbody>
</table>

| 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   | VREFBUF_CSR   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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>
<thead>
<tr>
<th>Reset value</th>
<th>VR</th>
<th>VRS</th>
<th>HIZ</th>
<th>ENVR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Refer to Section 2.2 for the register boundary addresses.
18 Comparator (COMP)

18.1 Introduction

The 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.

18.2 COMP main features

Each comparator has configurable plus and minus inputs used for flexible voltage selection:
- Multiplexed I/O pins
- 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)
18.3 COMP functional description

18.3.1 COMP block diagram

The block diagram of the comparators is shown in Figure 92.

![Figure 92. Comparator block diagram](MS47272V1)

18.3.2 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 OCREF_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

<table>
<thead>
<tr>
<th>Table 94. COMP1 input plus assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP1_INP</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>PC5 (available only on STM32WB55xx)</td>
</tr>
<tr>
<td>PB2</td>
</tr>
<tr>
<td>PA1</td>
</tr>
</tbody>
</table>
Table 95. COMP1 input minus assignment

<table>
<thead>
<tr>
<th>COMP1_INM</th>
<th>COMP1_INMSEL[2:0]</th>
<th>COMP1_INMESEL[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>¼ VREFINT</td>
<td>000</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>½ VREFINT</td>
<td>001</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>¾ VREFINT</td>
<td>010</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>VREFINT</td>
<td>011</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>Reserved</td>
<td>100</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>Reserved</td>
<td>101</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>PA9</td>
<td>110</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>PC4 (available only on STM32WB55xx)</td>
<td>111</td>
<td>00</td>
</tr>
<tr>
<td>PA0</td>
<td>111</td>
<td>01</td>
</tr>
<tr>
<td>PA4</td>
<td>111</td>
<td>10</td>
</tr>
<tr>
<td>PA5</td>
<td>111</td>
<td>11</td>
</tr>
</tbody>
</table>

1. N.A.: not affected.

Table 96. COMP2 input plus assignment

<table>
<thead>
<tr>
<th>COMP2_INP</th>
<th>COMP2_INPSEL</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>
</tbody>
</table>

Table 97. COMP2 input minus assignment

<table>
<thead>
<tr>
<th>COMP2_INM</th>
<th>COMP2_INMSEL[2:0]</th>
<th>COMP2_INMESEL[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>¼ VREFINT</td>
<td>000</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>½ VREFINT</td>
<td>001</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>¾ VREFINT</td>
<td>010</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>VREFINT</td>
<td>011</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>Reserved</td>
<td>100</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>Reserved</td>
<td>101</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>PB3</td>
<td>110</td>
<td>N.A.(1)</td>
</tr>
<tr>
<td>PB7</td>
<td>111</td>
<td>00</td>
</tr>
<tr>
<td>PA2</td>
<td>111</td>
<td>01</td>
</tr>
<tr>
<td>PA4</td>
<td>111</td>
<td>10</td>
</tr>
<tr>
<td>PA5</td>
<td>111</td>
<td>11</td>
</tr>
</tbody>
</table>

1. N.A.: not affected.
18.3.3 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.

**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.

18.3.4 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.

18.3.5 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.
18.3.6 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.

Figure 94. Comparator hysteresis
18.3.7 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.

![Figure 95. Comparator output blanking](image)

18.3.8 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 or 10: Medium speed / medium power
- 11: Low speed / ultra-low-power
18.4 COMP low-power modes

Table 98. Comparator behavior in the 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. Comparator interrupts cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Stop 1</td>
<td></td>
</tr>
<tr>
<td>Stop 2</td>
<td></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>

18.5 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 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 COMPx.

Table 99. Interrupt control bits

<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>
18.6  COMP registers

18.6.1  Comparator 1 control and status register (COMP1_CSR)

The COMP1_CSR is the Comparator 1 control/status register. It contains all the bits/flags related to comparator 1.

Address offset: 0x00

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>rs</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td>1 rw</td>
<td></td>
<td>1 rw</td>
<td></td>
<td>1 rw</td>
<td></td>
<td>1 rw</td>
<td></td>
<td>1 rw</td>
<td></td>
<td>1 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></td>
<td></td>
<td></td>
<td></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  LOCK: COMP1_CSR register lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator 1 control register, COMP1_CSR[31:0].

0: COMP1_CSR[31:0] for comparator 1 are read/write
1: COMP1_CSR[31:0] for comparator 1 are read-only

Bit 30  VALUE: Comparator 1 output status bit

This bit is read-only. It reflects the current comparator 1 output taking into account POLARITY bit effect.

Bits 29:27  Reserved, must be kept at reset value.

Bits 26:25  INMESEL[1:0]: Comparator 1 input minus extended selection bits.

These bits are set and cleared by software (only if LOCK is not set). They select which extended GPIO input is connected to the input minus of comparator if INMSEL = 111.

00: PC4 (available only on STM32WB55xx)
01: PA0
10: PA4
11: PA5

Bit 24  Reserved, must be kept at reset value.

Bit 23  SCAL EN: Voltage scaler enable bit

This bit is set and cleared by software. This bit enable the outputs of the VREFINT divider available on the minus input of the Comparator 1.

0: Bandgap scaler disable (if SCAL EN bit of COMP2_CSR register is also reset)
1: Bandgap scaler enable
**Bit 22 BRGEN: Scaler bridge enable**

This bit is set and cleared by software (only if LOCK not set). This bit enable the bridge of the scaler.
0: Scaler resistor bridge disable (if BRGEN bit of COMP2_CSR register is also reset)
1: Scaler resistor bridge enable

If SCALEN is set and BRGEN is reset, BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP.

If SCALEN and BRGEN are set, 1/4 BGAP, 1/2 BGAP, 3/4 BGAP and BGAP voltage references are available.

**Bit 21** Reserved, must be kept at reset value.

**Bits 20:18 BLANKING[2:0]: Comparator 1 blanking source selection bits**

These bits select which timer output controls the comparator 1 output blanking.
000: No blanking
001: TIM1 OC5 selected asblanking source
010: TIM2 OC3 selected as blanking source
All other values: reserved

**Bits 17:16 HYST[1:0]: Comparator 1 hysteresis selection bits**

These bits are set and cleared by software (only if LOCK not set). They select the hysteresis voltage of the comparator 1.
00: No hysteresis
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

**Bit 15 POLARITY: Comparator 1 polarity selection bit**

This bit is set and cleared by software (only if LOCK not set). It inverts Comparator 1 polarity.
0: Comparator 1 output value not inverted
1: Comparator 1 output value inverted

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

**Bits 8:7 INPSEL[1:0]: Comparator 1 input plus selection bit**

This bit is set and cleared by software (only if LOCK not set).
00: External I/O - PC5 (available only on STM32WB55xx)
01: PB2
10: PA1
11: Reserved

**Bits 6:4 INMSEL[2:0]: Comparator 1 input minus selection bits**

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator 1.
000 = 1/4 VREFINT
001 = 1/2 VREFINT
010 = 3/4 VREFINT
011 = VREFINT
100 = Reserved
101 = Reserved
110 = PA9
111 = GPIOx selected by INMESEL bits
Bits 3:2  **PWRMODE[1:0]:** Power Mode of the comparator 1
These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator 1.
- 00: High speed
- 01 or 10: Medium speed
- 11: Ultra low power

Bit 1  Reserved, must be kept at reset value.

Bit 0  **EN:** Comparator 1 enable bit
This bit is set and cleared by software (only if LOCK not set). It switches on Comparator 1.
- 0: Comparator 1 switched OFF
- 1: Comparator 1 switched ON

### 18.6.2 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 comparator 2.

**Address offset:** 0x04

**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>rs</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>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>
<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>

Bit 31  **LOCK:** CSR register lock bit
This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator 2 control register, COMP2_CSR[31:0].
- 0: COMP2_CSR[31:0] for comparator 2 are read/write
- 1: COMP2_CSR[31:0] for comparator 2 are read-only

Bit 30  **VALUE:** Comparator 2 output status bit
This bit is read-only. It reflects the current comparator 2 output taking into account POLARITY bit effect.

Bits 29:27  Reserved, must be kept at reset value.

Bits 26:25  **INMESEL[1:0]:** comparator 2 input minus extended selection bits.
These bits are set and cleared by software (only if LOCK is not set). They select which extended GPIO input is connected to the input minus of comparator if INMSEL = 111.
- 00: PB7
- 01: PA2
- 10: PA4
- 11: PA5

Bit 24  Reserved, must be kept at reset value.
Bit 23  **SCALEN**: Voltage scaler enable bit  
This bit is set and cleared by software. This bit enable the outputs of the $V_{\text{REFINT}}$ divider available on the minus input of the Comparator 2.  
0: Bandgap scaler disable (if SCALEN bit of COMP1_CSR register is also reset)  
1: Bandgap scaler enable  

Bit 22  **BRGEN**: Scaler bridge enable  
This bit is set and cleared by software (only if LOCK not set). This bit enable the bridge of the scaler.  
0: Scaler resistor bridge disable (if BRGEN bit of COMP1_CSR register is also reset)  
1: Scaler resistor bridge enable  
If SCALEN is set and BRGEN is reset, BG voltage reference is available but not 1/4 BGAP, 1/2 BGAP, 3/4 BGAP. BGAP value is sent instead of 1/4 BGAP, 1/2 BGAP, 3/4 BGAP.  
If SCALEN and BRGEN are set, 1/4 BGAP 1/2 BGAP 3/4 BGAP and BGAP voltage references are available.  

Bit 21  Reserved, must be kept at reset value.  

Bits 20:18  **BLANKING[2:0]**: Comparator 2 blanking source selection bits  
These bits select which timer output controls the comparator 2 output blanking.  
000: No blanking  
001: TIM1 OC5 selected as blanking source  
010: TIM2 OC3 selected as blanking source  
All other values: reserved  

Bits 17:16  **HYST[1:0]**: Comparator 2 hysteresis selection bits  
These bits are set and cleared by software (only if LOCK not set). Select the hysteresis voltage of the comparator 2.  
00: No hysteresis  
01: Low hysteresis  
10: Medium hysteresis  
11: High hysteresis  

Bit 15  **POLARITY**: Comparator 2 polarity selection bit  
This bit is set and cleared by software (only if LOCK not set). It inverts Comparator 2 polarity.  
0: Comparator 2 output value not inverted  
1: Comparator 2 output value inverted  

Bits 14:10  Reserved, must be kept at reset value.  

Bit 9  **WINMODE**: Windows mode selection bit  
This bit is set and cleared by software (only if LOCK not set). This bit selects the window mode of the comparators. If set, both positive inputs of comparators are connected together.  
0: Input plus of Comparator 2 is not connected to Comparator 1  
1: Input plus of Comparator 2 is connected with input plus of Comparator 1  

Bits 8:7  **INPSEL[1:0]**: Comparator 2 input plus selection bit  
This bit is set and cleared by software (only if LOCK not set).  
00: PB4  
01: PB6  
10: PA3  
11: Reserved
Bits 6:4  **INMSEL[2:0]**: Comparator 2 input minus selection bits
These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator 2.
- 000 = 1/4 VREFINT
- 001 = 1/2 VREFINT
- 010 = 3/4 VREFINT
- 011 = VREFINT
- 100 = Reserved
- 101 = Reserved
- 110 = PB3
- 111 = GPIOx selected by INMSEL bits

Bits 3:2  **PWRMODE[1:0]**: Power Mode of the comparator 2
These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator 2.
- 00: High speed
- 01 or 10: Medium speed
- 11: Ultra low power

Bit 1  Reserved, must be kept at reset value.

Bit 0  **EN**: Comparator 2 enable bit
This bit is set and cleared by software (only if LOCK not set). It switches on comparator 2.
- 0: Comparator 2 switched OFF
- 1: Comparator 2 switched ON
### 18.6.3 COMP register map

The following table summarizes the comparator registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td>0x00</td>
<td>COMP1_CSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOCK</td>
<td>VALUE</td>
<td>INMSEL</td>
<td>SCALEN</td>
<td>BRGEN</td>
<td>BLANKING</td>
<td>HYST</td>
<td>POLARITY</td>
<td>POLARITY</td>
<td>WINMODE</td>
<td>INPSEL</td>
<td>INMSEL</td>
<td>PW_MODE</td>
<td>PW_MODE</td>
<td>PW_MODE</td>
<td>PW_MODE</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td>EN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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>
<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>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.2 on page 66 for the register boundary addresses.
19 Liquid crystal display controller (LCD)

This section applies to STM32WB55xx devices only.

19.1 LCD introduction

The LCD controller is a digital controller/driver for monochrome passive liquid crystal display (LCD) with up to eight common terminals, and up to 44 segment terminals to drive 176 (44x4) or 320 (40x8) LCD picture elements (pixels). The exact number of terminals depends on the device pinout as described in the datasheet.

The LCD is made up of several segments (pixels or complete symbols) that can be turned visible or invisible. Each segment consists of a layer of liquid crystal molecules aligned between two electrodes. When a voltage greater than a threshold voltage is applied across the liquid crystal, the segment becomes visible. The segment voltage must be alternated to avoid an electrophoresis effect in the liquid crystal (which degrades the display). The waveform across a segment must then be generated, so to avoid having a direct current (DC).

Glossary

Bias: number of voltage levels used when driving an LCD
It is defined as \( \frac{1}{\text{number of voltage levels used to drive an LCD display - 1}} \).

Boost circuit: contrast controller circuit

Common: electrical connection terminal connected to several segments (44 segments)

Duty ratio: number defined as \( \frac{1}{\text{number of common terminals on a given LCD display}} \)

Frame: one period of the waveform written to a segment

Frame rate: number of frames per second (number of times the LCD segments are energized per second)

LCD (liquid crystal display): passive display panel with terminals leading directly to a segment

Segment: smallest viewing element (a single bar or dot that is used to help create a character on an LCD display)

19.2 LCD main features

- Highly flexible frame rate control
- Static, 1/2, 1/3, 1/4, and 1/8 duty supported
- Static, 1/2, 1/3, and 1/4 bias supported
- Double-buffered memory allowing data in LCD_RAM registers to be updated at any time by the application firmware without affecting the integrity of the data displayed
  - LCD data RAM of up to 16 x 32-bit registers which contain pixel information (active/inactive)
- Software selectable LCD output voltage (contrast) from \( V_{LCD_{min}} \) to \( V_{LCD_{max}} \)
No need for external analog components:
- stepup converter embedded to generate an internal $V_{\text{LCD}}$ voltage higher than $V_{\text{DD}}$
- software selection between external and internal $V_{\text{LCD}}$ voltage source. In case of an external source, the internal boost circuit is disabled to reduce power consumption.
- resistive network embedded to generate intermediate $V_{\text{LCD}}$ voltages
- structure of the resistive network configurable by software to adapt the power consumption to match the capacitive charge required by the LCD panel
- integrated voltage output buffers for higher LCD driving capability.

Contrast that can be adjusted using two different methods:
- When using the internal step-up converter, the software can adjust $V_{\text{LCD}}$ between $V_{\text{LCDmin}}$ and $V_{\text{LCDmax}}$. 
- Programmable dead time (up to eight phase periods) between frames

Full support of low-power modes: The LCD controller can be displayed in Sleep, Low-power run, Low-power sleep, and Stop modes, or can be fully disabled to reduce power consumption.

Built-in phase inversion for reduced power consumption and EMI (electromagnetic interference)

Start-of-frame interrupt to synchronize the software when updating the LCD data RAM

Blink capability:
- Up to 1, 2, 3, 4, 8, or all pixels which can be programmed to blink at a configurable frequency
- Software adjustable blink frequency to achieve around 0.5 Hz, 1 Hz, 2 Hz, or 4 Hz

Used LCD segment and common pins must be configured as GPIO alternate functions, and unused segment and common pins can be used for GPIO or for another peripheral alternate function.

Note: When the LCD relies on the internal stepup converter, the $VLCD$ pin must be connected to $V_{SS}$ with a capacitor. Its typical value is 1 $\mu$F (see $C_{\text{EXT}}$ value in the datasheet for details).
19.3 LCD functional description

19.3.1 General description

The LCD controller has five main blocks (see Figure 96):

Figure 96. LCD controller block diagram

Note: LCDCLK is the same as RTCCLK. Refer to the RTC/LCD clock description in the RCC section of the reference manual.

The frequency generator allows the user to achieve various LCD frame rates starting from an LCD input clock frequency (LCDCLK) which can vary from 32 kHz up to 1 MHz.

Three different clock sources can be used to provide the LCD clock (LCDCLK/RTCCLK):

- 32 kHz low-speed external RC (LSE)
- 32 kHz low-speed internal RC (LSI)
- high-speed external (HSE) divided by 32
19.3.2 Frequency generator

This clock source must be stable in order to obtain accurate LCD timing, and hence to minimize DC voltage offset across LCD segments. The input clock LCDCLK can be divided by any value from 1 to \(2^{15} \times 31\) (see Section 19.6.2). The frequency generator consists of a prescaler (16-bit ripple counter), and a 16 to 31 clock divider. PS\([3:0]\) in LCD_FCR selects LCDCLK divided by \(2^{PS[3:0]}\). If a finer resolution rate is required, DIV\([3:0]\) in LCD_FCR can be used to divide the clock further by 16 to 31.

In this way, the user can roughly scale the frequency, and then fine-tune it by linearly scaling the clock with the counter. The output of the frequency generator block is \(f_{\text{ck}_\text{div}}\), which constitutes the time base for the entire LCD controller. The \(f_{\text{ck}_\text{div}}\) frequency is equivalent to the LCD phase frequency, rather than the frame frequency (they are equal only in case of static duty). The frame frequency \((f_{\text{frame}})\) is obtained from \(f_{\text{ck}_\text{div}}\) by dividing it by the number of active common terminals (or by multiplying it for the duty).

Thus the relation between the input clock frequency \((f_{\text{LCDCLK}})\) of the frequency generator and its output clock frequency \(f_{\text{ck}_\text{div}}\) is:

\[
f_{\text{ck}_\text{div}} = \frac{f_{\text{LCDCLK}}}{2^{\text{PS}} \times 16 + \text{DIV}}
\]

\[f_{\text{frame}} = f_{\text{ck}_\text{div}} \times \text{duty}\]

This makes the frequency generator very flexible. An example of frame rate calculation is shown in the table below.

<table>
<thead>
<tr>
<th>LCDCLK</th>
<th>PS([3:0])</th>
<th>DIV([3:0])</th>
<th>Ratio</th>
<th>Duty</th>
<th>(f_{\text{frame}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>32.768 kHz</td>
<td>3</td>
<td>1</td>
<td>136</td>
<td>1/8</td>
<td>30.12 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>4</td>
<td>1</td>
<td>272</td>
<td>1/4</td>
<td>30.12 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>4</td>
<td>6</td>
<td>352</td>
<td>1/3</td>
<td>31.03 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>5</td>
<td>1</td>
<td>544</td>
<td>1/2</td>
<td>30.12 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>6</td>
<td>1</td>
<td>1088</td>
<td>static</td>
<td>30.12 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>1</td>
<td>4</td>
<td>40</td>
<td>1/8</td>
<td>102.40 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>2</td>
<td>4</td>
<td>80</td>
<td>1/4</td>
<td>102.40 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>2</td>
<td>11</td>
<td>108</td>
<td>1/3</td>
<td>101.14 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>3</td>
<td>4</td>
<td>160</td>
<td>1/2</td>
<td>102.40 Hz</td>
</tr>
<tr>
<td>32.768 kHz</td>
<td>4</td>
<td>4</td>
<td>320</td>
<td>static</td>
<td>102.40 Hz</td>
</tr>
<tr>
<td>1.00 MHz</td>
<td>6</td>
<td>3</td>
<td>1216</td>
<td>1/8</td>
<td>102.80 Hz</td>
</tr>
<tr>
<td>1.00 MHz</td>
<td>7</td>
<td>3</td>
<td>2432</td>
<td>1/4</td>
<td>102.80 Hz</td>
</tr>
<tr>
<td>1.00 MHz</td>
<td>7</td>
<td>10</td>
<td>3328</td>
<td>1/3</td>
<td>100.16 Hz</td>
</tr>
<tr>
<td>1.00 MHz</td>
<td>8</td>
<td>3</td>
<td>4864</td>
<td>1/2</td>
<td>102.80 Hz</td>
</tr>
<tr>
<td>1.00 MHz</td>
<td>9</td>
<td>3</td>
<td>9728</td>
<td>static</td>
<td>102.80 Hz</td>
</tr>
</tbody>
</table>
The frame frequency must be selected to be within a range of around ~30 Hz to ~100 Hz. It is a compromise between power consumption and the acceptable refresh rate. In addition, a dedicated blink prescaler selects the blink frequency. This frequency is defined as:

\[ f_{\text{BLINK}} = f_{\text{ck_div/2}}^{(\text{BLINKF} + 3)} \]

with BLINKF[2:0] = 0, 1, 2, ..., 7

The blink frequency achieved is in the range of 0.5 Hz, 1 Hz, 2 Hz, or 4 Hz.

19.3.3 Common driver

Common signals are generated by the common driver block (see Figure 96).

COM signal bias

Each COM signal has identical waveforms, but different phases. It has its max amplitude \( V_{\text{LCD}} \) or \( V_{\text{SS}} \) only in the corresponding phase of a frame cycle, while during the other phases, the signal amplitude is:

- \( 1/4 \) \( V_{\text{LCD}} \) or \( 3/4 \) \( V_{\text{LCD}} \) in case of 1/4 bias
- \( 1/3 \) \( V_{\text{LCD}} \) or \( 2/3 \) \( V_{\text{LCD}} \) in case of 1/3 bias
- \( 1/2 \) \( V_{\text{LCD}} \) in case of 1/2 bias

The selection between 1/2, 1/3, and 1/4 bias mode can be done through BIAS[1:0] in LCD_CR.

A pixel is activated when both of its corresponding common and segment lines are active during the same phase: when the voltage difference between common and segment is maximum during this phase. Common signals are phase inverted in order to reduce EMI.

As shown in the figure below, with phase inversion, there is a mean voltage of \( 1/2 \) \( V_{\text{LCD}} \) at the end of every odd cycle.

In case of 1/2 bias (BIAS = 01), the VLCD pin generates an intermediate voltage equal to \( 1/2 \) \( V_{\text{LCD}} \) on node b for odd and even frames (see Figure 100).
COM signal duty

Depending on DUTY[2:0] in LCD_CR, the COM signals are generated with static duty (see Figure 99), 1/2 duty (see Figure 100), 1/3 duty (see Figure 101), 1/4 duty (see Figure 102), or 1/8 duty (see Figure 103).

COM[n] n[0 to 7] is active during phase n in the odd frame, so the COM pin is driven to VLCD.

During phase n of the even frame, the COM pin is driven to VSS.

In the case of 1/3 or 1/4) bias, COM[n] is inactive during phases other than n, so the COM pin is driven to 1/3 (1/4) VLCD during odd frames, and to 2/3 (3/4) VLCD during even frames.

In the case of 1/2 bias, if COM[n] is inactive during phases other than n, the COM pin is always driven (odd and even frame) to 1/2 VLCD.

When static duty is selected, the segment lines are not multiplexed, which means that each segment output corresponds to one pixel. In this way, only up to 44 pixels can be driven.

COM[0] is always active while COM[7:1] are not used and are driven to VSS.

When LCDEN in LCD_CR is reset, all common lines are pulled down to VSS, and the ENS flag in LCD_SR becomes 0. Static duty means that COM[0] is always active, and only two voltage levels are used for segment and common lines: VLCD and VSS. A pixel is active if the corresponding SEG line has a voltage opposite to that of the COM, and inactive when the voltages are equal. In this way the LCD has maximum contrast (see Figure 98, Figure 99).

In the figure below, pixel 0 is active while pixel 1 is inactive.

**Figure 98. Static duty case 1**

In each frame, there is only one phase, this is why f_frame is equal to f_LCD. If 1/4 duty is selected there are four phases in a frame in which COM[0] is active during phase 0, COM[1] is active during phase 1, COM[2] is active during phase 2, and COM[3] is active during phase 3.
In this mode, the segment terminals are multiplexed, and each of them control four pixels. A pixel is activated only when both of its corresponding SEG and COM lines are active in the same phase. In case of 1/4 duty, to deactivate pixel 0 connected to COM[0], SEG[0] needs to be inactive during the phase 0 when COM[0] is active. To activate pixel 0 connected to COM[1], SEG[0] needs to be active during phase 1 when COM[1] is active (see Figure 102). To activate pixels from 0 to 43 connected to COM[0], SEG[0:43] needs to be active during phase 0 when COM[0] is active. These considerations can be extended to the other pixels.

**Eight-to-one mux**

When COM[0] is active the common driver block, it also drives the eight-to-one mux shown in Figure 96 in order to select the content of first two RAM register locations. When COM[7] is active, the output of the eight-to-one mux is the content of the last two RAM locations.
19.3.4 Segment driver

The segment driver block controls the SEG lines according to the pixel data coming from the eight-to-one mux driven in each phase by the common driver block.

**In the case of 1/4 or 1/8 duty**

When COM[0] is active, the pixel information (active/inactive) related to the pixel connected to COM[0] (content of the first two LCD_RAM locations) goes through the eight-to-one mux.

The SEG[n] pin [0 to 43] is driven to VSS (indicating pixel n is active when COM[0] is active) in phase 0 of the odd frame.

The SEG[n] pin is driven to V_LCD in phase 0 of the even frame. If pixel n is inactive then the SEG[n] pin is driven to 2/3 (2/4) V_LCD in the odd frame or 1/3 (2/4) V_LCD in the even frame (current inversion in V_LCD pad) (see Figure 97).

In case of 1/2 bias, if the pixel is inactive the SEG[n] pin is driven to V_LCD in the odd and to V_SS in the even frame.

When the LCD controller is disabled (LCDEN cleared in LCD_CR), then the SEG lines are pulled down to V_SS.
Figure 101. 1/3 duty, 1/3 bias
Figure 102. 1/4 duty, 1/3 bias

Liquid crystal display and terminal connection

Selected waveform

Non selected waveform

1 frame
Figure 103. 1/8 duty, 1/4 bias

Liquid crystal display and terminal connection

- PIN COM0
- PIN COM1
- PIN COM2
- PIN COM7
- PIN SEG0

COM0-SEG0 selected waveform

COM2-SEG0 non selected waveform

1 frame
**Blink**

The segment driver also implements a programmable blink feature to allow some pixels to continuously switch on at a specific frequency. The blink mode can be configured by BLINK[1:0] in LCD_FCR, making possible to blink up to 1, 2, 4, 8, or all pixels (see Section 19.6.2). The blink frequency can be selected from eight different values using BLINKF[2:0] in LCD_FCR.

The table below gives examples of different blink frequencies (as a function of ck_div frequency).

<table>
<thead>
<tr>
<th>BLINKF[2:0]</th>
<th>ck_div frequency (with LCDCLK frequency of 32.768 kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32 Hz</td>
</tr>
<tr>
<td>0 0 0</td>
<td>4.0 Hz</td>
</tr>
<tr>
<td>0 0 1</td>
<td>2.0 Hz</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1.0 Hz</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0.5 Hz</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0.25 Hz</td>
</tr>
<tr>
<td>1 0 1</td>
<td>N/A</td>
</tr>
<tr>
<td>1 1 0</td>
<td>N/A</td>
</tr>
<tr>
<td>1 1 1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**19.3.5 Voltage generator and contrast control**

**LCD supply source**

The LCD power supply source comes from either the internal stepup converter, or from an external voltage applied on the VLCD pin. The internal or external voltage source can be selected using VSEL in LCD_CR. In case of external source selected, the internal boost circuit (stepup converter) is disabled to reduce power consumption.

When the stepup converter is selected as V_LCD source, the V_LCD value can be chosen among a wide set of values from V_LCDmin to V_LCDmax by means of CC[2:0] (contrast control) in LCD_FCR (see Section 19.6.2). New values of V_LCD takes effect every beginning of a new frame.

When external power source is selected as V_LCD source, the V_LCD voltage must be chosen in the range of V_LCDmin to V_LCDmax (see datasheets). The contrast can then be controlled by programming a dead time between frames (see Deadtime).

A specific software sequence must be performed to configure the LCD depending on the LCD power supply source to be used. The LCD controller is considered as disabled prior to the configuration sequence.

In case the internal stepup converter is used (capacitor C_EXT on VLCD pin is required):

- Configure the VLCD pin as alternate function LCD in the GPIO_AFR register.
- Wait for the external capacitor C_EXT to be charged (C_EXT connected to the VLCD pin, approximately 2 ms for C_EXT = 1 μF)
Set voltage source to internal source by resetting VSEL in LCD_CR.
Enable the LCD controller by setting LCDEN in LCD_CR.

In case of LCD external power source is used:
Set voltage source to external source by setting VSEL in LCD_CR.
Configure the VLCD pin as alternate function LCD in GPIO_AFR.
Enable the LCD controller by setting LCDEN in LCD_CR.

**LCD intermediate voltages**

The LCD intermediate voltage levels are generated through an internal resistor divider network as shown in Figure 104.

The LCD voltage generator issues intermediate voltage levels between V_{SS} and V_{LCD}:
- 1/3 V_{LCD} and 2/3 V_{LCD} in case of 1/3 bias
- 1/4 V_{LCD}, 2/4 V_{LCD} and 3/4 V_{LCD} in case of 1/4 bias
- only 1/2 V_{LCD} in case of 1/2 bias.

**LCD drive selection**

Two resistive networks, one with low-value resistors (R_L), and one with high-value resistors (R_H) are respectively used to increase the current during transitions, and to reduce power consumption in static state.

The EN switch follows the rules described below (see Figure 104):
- If LCDEN is set in LCD_CR, the EN switch is closed.
- When clearing LCDEN in LCD_CR, the EN switch is open at the end of the even frame in order to avoid a medium voltage level different from 0 considering the entire frame odd plus even.

PON[2:0] (pulse on duration) in LCD_FCR configures the time during which R_L is enabled through the HD (high drive) switch when the levels of the common and segment lines change (see Figure 104). A short drive time leads to lower power consumption, but displays with high internal resistance may need a longer drive time to achieve satisfactory contrast.
The \( R_{LN} \) and \( R_{HN} \) are the low value resistance network and the high value resistance network, respectively.

The HD switch follows the rules described below:

- If HD and PON[2:0] are reset in LCD_FCR, the HD switch is open.
- If HD is reset in LCD_FCR, and PON[2:0] in LCD_FCR are different from 00, the HD switch is closed during the number of pulses defined in PON[2:0].
- If HD = 1 in LCD_FCR, then HD switch is always closed.

After LCDEN is activated, RDY is set in LCD_SR to indicate that voltage levels are stable and the LCD controller can start to work.

**Buffered mode**

When voltage output buffers are enabled by setting BUFEN in LCD_CR, the LCD driving capability is improved as buffers prevent the LCD capacitive loads from loading the resistor bridge unacceptably and interfering with its voltage generation. As a result, intermediate voltage levels are more stable, which improves RMS voltage applied to the LCD pixels.

In buffer mode, intermediate voltages are generated by the high value resistor bridge \( R_{HN} \) to reduce power consumption. The low value resistor bridge \( R_{LN} \) is automatically disabled whatever HD or PON[2:0] configuration.
Buffers can be used independently of the $V_{LCD}$ supply source (internal or external), but they can only be enabled or disabled when the LCD controller is not activated.

**Deadtime**

In addition to using $CC[2:0]$, the contrast can be controlled by programming a dead time between each frame. During the dead time, the COM and SEG values are put to $V_{SS}$. DEAD[2:0] in LCD_FCR can be used to program a time of up to eight phase periods. This dead time reduces the contrast without modifying the frame rate.

**Figure 105. Deadtime**

19.3.6 Double-buffer memory

Using its double-buffer memory, the LCD controller ensures the coherency of the displayed information without having to use interrupts to control the LCD_RAM modification.

The application software can access the first buffer level (LCD_RAM) through the APB interface. Once it has modified the LCD_RAM, it sets UDR in LCD_SR. This UDR flag (update display request) requests the updated information to be moved into the second buffer level (LCD_DISPLAY).

This operation is done synchronously with the frame (at the beginning of the next frame). Until the update is completed, the LCD_RAM is write protected, and the UDR flag stays high. Once the update is completed, another flag (UDD, update display done) is set and generates an interrupt if UDDIE is set in LCD_FCR.

The time it takes to update LCD_DISPLAY is, in the worst case, one odd and one even frame.

The update does not occur (UDR = 1 and UDD = 0) until the display is enabled (LCDEN = 1)

19.3.7 COM and SEG multiplexing

**Output pins versus duty modes**

The output pins consists of:

- SEG[43:0]
- COM[3:0]
Depending on the duty configuration, the COM and SEG output pins can have different functions:

- In static, 1/2, 1/3, and 1/4 duty modes, there are up to 44 SEG pins and respectively 1, 2, 3, and 4 COM pins.
- In 1/8 duty mode (DUTY[2:0] = 100), COM[7:4] outputs are available on SEG[43:40] pins, reducing to the number of available segments to 40.

**Remapping capability for small packages**

Additionally, it is possible to remap four segments by setting MUX_SEG in LCD_CR. This is particularly useful when using smaller device types with fewer external pins.

When MUX_SEG is set, output pins SEG[43:40] have the same function as SEG[31:28].

This feature is available only if the mode 1/8 duty is not selected.

Check the availability of this feature by referring to the pinout section of the datasheet.

For the considered package, check the availability of the SEG/COM multiplexing pin as follow (n = number of segment for the considered package):

- LCD_SEG[n-1]/LCD_COM7/LCD_SEG[31]
- LCD_SEG[n-2]/LCD_COM6/LCD_SEG[30]
- LCD_SEG[n-3]/LCD_COM5/LCD_SEG[29]
- LCD_SEG[n-4]/LCD_COM4/LCD_SEG[28]

**Summary of COM and SEG functions versus duty and remap**

All the possible ways of multiplexing the COM and SEG functions are described in the table below.

<table>
<thead>
<tr>
<th>DUTY</th>
<th>MUX_SEG</th>
<th>WLCSP100</th>
<th>VFQFPN68</th>
<th>UFQFPN48</th>
<th>Output pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>N/A</td>
<td>40x8</td>
<td>-</td>
<td>-</td>
<td>COM[3:0]</td>
<td>COM[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[39:0]</td>
<td>SEG[39:0]</td>
</tr>
<tr>
<td>DUTY</td>
<td>MUX_SEG</td>
<td>WLCSP100</td>
<td>VFQFPN68</td>
<td>UFQFPN48</td>
<td>Output pin</td>
<td>Function</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>0</td>
<td>44x4</td>
<td>-</td>
<td>-</td>
<td></td>
<td>COM[3:0]</td>
<td>COM[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[39:0]</td>
<td>SEG[39:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[3:0]</td>
<td>COM[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[31:28]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[27:0]</td>
<td>SEG[27:0]</td>
</tr>
<tr>
<td>1/4</td>
<td>40x4</td>
<td>-</td>
<td>-</td>
<td></td>
<td>COM[3:0]</td>
<td>COM[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td>SEG[24:0]</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>28x4</td>
<td>-</td>
<td></td>
<td>COM[3:0]</td>
<td>COM[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td>SEG[24:0]</td>
</tr>
<tr>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>13x4</td>
<td></td>
<td>COM[3:0]</td>
<td>COM[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[21]</td>
<td>SEG[21]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[17:16]</td>
<td>SEG[17:16]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[9:0]</td>
<td>SEG[9:0]</td>
</tr>
<tr>
<td>DUTY</td>
<td>MUX_SEG</td>
<td>WLCSP100</td>
<td>VFQFPN68</td>
<td>UFQFPN48</td>
<td>Output pin</td>
<td>Function</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>-------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>1/3</td>
<td>0</td>
<td>44x3</td>
<td>-</td>
<td>-</td>
<td>COM[3]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[2:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[39:0]</td>
<td>SEG[39:0]</td>
</tr>
<tr>
<td>1/3</td>
<td>1</td>
<td>40x3</td>
<td>-</td>
<td>-</td>
<td>COM[3]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[2:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[31:28]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[27:0]</td>
<td>SEG[27:0]</td>
</tr>
<tr>
<td>1/3</td>
<td>0</td>
<td>-</td>
<td>28x3</td>
<td>-</td>
<td>COM[3]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[2:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td>SEG[24:0]</td>
</tr>
<tr>
<td>1/3</td>
<td>1</td>
<td>-</td>
<td>28x3</td>
<td>-</td>
<td>COM[3]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[2:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td>SEG[24:0]</td>
</tr>
<tr>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>13x3</td>
<td></td>
<td>COM[3]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[2:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[21]</td>
<td>SEG[21]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[17:16]</td>
<td>SEG[17:16]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[9:0]</td>
<td>SEG[9:0]</td>
</tr>
<tr>
<td>1/2</td>
<td>0</td>
<td>44x2</td>
<td>-</td>
<td>-</td>
<td>COM[3:2]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[1:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[39:0]</td>
<td>SEG[39:0]</td>
</tr>
<tr>
<td>1/2</td>
<td>1</td>
<td>40x2</td>
<td>-</td>
<td>-</td>
<td>COM[3:2]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[1:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[31:28]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[27:0]</td>
<td>SEG[27:0]</td>
</tr>
</tbody>
</table>

Table 103. Remapping capability (continued)
<table>
<thead>
<tr>
<th>DUTY</th>
<th>MUX_SEG</th>
<th>WLCSP100</th>
<th>VFQFPN68</th>
<th>UFQFPN48</th>
<th>Output pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>28x2</td>
<td>-</td>
<td></td>
<td>COM[3:2]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[1:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td></td>
</tr>
<tr>
<td>1/2</td>
<td>-</td>
<td>28x2</td>
<td>-</td>
<td></td>
<td>COM[3:2]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[1:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>13x2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[3:1]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[39:0]</td>
<td></td>
</tr>
<tr>
<td>STATIC</td>
<td>0</td>
<td>44x1</td>
<td>-</td>
<td>-</td>
<td>COM[3:1]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[31:28]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[27:0]</td>
<td>SEG[27:0]</td>
</tr>
<tr>
<td>STATIC</td>
<td>1</td>
<td>40x1</td>
<td>-</td>
<td>-</td>
<td>COM[3:1]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[31:28]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[27:0]</td>
<td>SEG[27:0]</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>28x1</td>
<td>-</td>
<td></td>
<td>COM[3:1]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td>SEG[24:0]</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>28x1</td>
<td>-</td>
<td></td>
<td>COM[3:1]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[24:0]</td>
<td>SEG[24:0]</td>
</tr>
</tbody>
</table>
The figure below gives examples showing the signal connections to the external pins.

**Figure 106. SEG/COM mux feature example**

<table>
<thead>
<tr>
<th>DUTY</th>
<th>MUX_SEG</th>
<th>WLCSP100</th>
<th>VFQFPN68</th>
<th>UFQFPN48</th>
<th>Output pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATIC</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>13x1</td>
<td>COM[3:1]</td>
<td>not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COM[0]</td>
<td>COM[0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[21]</td>
<td>SEG[21]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[17:16]</td>
<td>SEG[17:16]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SEG[9:0]</td>
<td>SEG[9:0]</td>
</tr>
</tbody>
</table>
### 19.3.8 Flowchart

**Figure 107. Flowchart example**

**START**

- **INIT**
  - Enable the GPIO port clocks
  - Configure the LCD GPIO pins as alternate functions
  - Configure LCD controller according to the display to be driven

- Load the initial data to be displayed into **LCD_RAM** and set the **UDR** in **LCD_SR**

- Program the desired frame rate (**PS** and **DIV** in **LCD_FCR**)
- Program the contrast (**CC** in **LCD_FCR**)

Enable the display (**LCDEN** in **LCD_CR**)

**Adjust contrast?**
  - Yes: Change **PS**, **DIV**, **CC**, **PON**, **DEAD**, or **HD** in **LCD_FCR**
  - No: Modify data?
    - Yes: **UDR** = 1?
      - Yes: Modify the **LCD_RAM**
      - No: Set **UDR** in **LCD_SR**
    - No: Change blink?
      - Yes: Change **BLINK** or **BLINKF** in **LCD_FCR**
      - No: Disable LCD?
        - Yes: Disable the display (**LCDEN** in **LCD_CR**)
        - No: **ENS** = 0?
          - Yes: END
          - No: **ENS** = 0?
19.4 LCD low-power modes

The LCD controller can be displayed in Stop mode, or can be fully disabled to reduce power consumption.

19.5 LCD interrupts

The table below gives the list of LCD interrupt requests.

Table 104. LCD interrupt requests

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event flag clearing method</th>
<th>Interrupt enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of frame (SOF)</td>
<td>SOF</td>
<td>Write SOFC = 1</td>
<td>SOFIE</td>
</tr>
<tr>
<td>Update display done (UDD)</td>
<td>UDD</td>
<td>Write UDDC = 1</td>
<td>UDDIE</td>
</tr>
</tbody>
</table>

**Start of frame (SOF)**

The LCD start of frame interrupt is executed if SOFIE (start of frame interrupt enable) is set in LCD_FCR. SOF is cleared by writing SOFC to 1 in LCD_CLR when executing the corresponding interrupt handling vector.

**Update display done (UDD)**

The LCD update display interrupt is executed if UDDIE (update display done interrupt enable) is set in LCD_FCR. UDD is cleared by writing UDDC to 1 in LCD_CLR when executing the corresponding interrupt handling vector.

Depending on the product implementation, all these interrupts can either share the same interrupt vector (LCD global interrupt), or be grouped into two interrupt vectors (LCD SOF interrupt and LCD UDD interrupt). Refer to the Table 60: CPU1 vector table for details.

To enable the LCD interrupts, the following sequence is required:

1. Configure and enable the LCD IRQ channel in the NVIC.
2. Configure the LCD to generate interrupts.
19.6 LCD registers

These registers are accessed by words (32-bit).

19.6.1 LCD control register (LCD_CR)

Address offset: 0x00
Reset value: 0x0000 0000

VSEL, MUX_SEG, BIAS, DUTY, and BUFEN bitfields are write-protected when the LCD is enabled (ENS =1 in LCD_SR).

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

Bit 8 **BUFEN**: Voltage output buffer enable
This bit is used to enable/disable the voltage output buffer for higher driving capability.
0: Output buffer disabled
1: Output buffer enabled

Bit 7 **MUX_SEG**: Mux segment enable
This bit is used to enable SEG pin remapping. Four SEG pins can be multiplexed with SEG[31:28]. See Section 19.3.7.
0: SEG pin multiplexing disabled
1: SEG[31:28] multiplexed with SEG[43:40]

Bits 6:5 **BIAS[1:0]**: Bias selector
These bits determine the bias used. Value 11 is forbidden.
00: Bias 1/4
01: Bias 1/2
10: Bias 1/3
11: Reserved

Bits 4:2 **DUTY[2:0]**: Duty selection
These bits determine the duty cycle. Values 101, 110 and 111 are forbidden.
000: Static duty
001: 1/2 duty
010: 1/3 duty
011: 1/4 duty
100: 1/8 duty
Others: Reserved

Bit 1 **VSEL**: Voltage source selection
This bit determines the voltage source for the LCD.
0: Internal source (voltage stepup converter)
1: External source (VLCD pin)
Bit 0  **LCDEN**: LCD controller enable
   This bit is set by software to enable the LCD controller/driver. It is cleared by software to turn off the LCD at the beginning of the next frame. When the LCD is disabled, all COM and SEG pins are driven to VSS.
   0: LCD controller disabled
   1: LCD controller enabled

### 19.6.2 LCD frame control register (LCD_FCR)

Address offset: 0x04
Reset value: 0x0000 0000

This register can be updated any time, but the new values are applied only at the beginning of the next frame (except for UDDIE, SOFIE that affect the device behavior immediately). The new value of CC[2:0] is also applied immediately, but its effect on the device is delayed at the beginning of the next frame by the voltage generator.

Reading this register gives the last value written in the register, and not the configuration used to display the current frame.

When BUFEN is set in LCD_CR, the low resistor divider network is automatically disabled whatever HD or PON[2:0] configuration.

<table>
<thead>
<tr>
<th>Bit 31:26 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 25:22 <strong>PS[3:0]</strong>: PS 16-bit prescaler</td>
</tr>
<tr>
<td>These bits are written by software to define the division factor of the PS 16-bit prescaler.</td>
</tr>
<tr>
<td>( \text{ck}_{ps} = \frac{\text{LCDCLK}}{2^{\text{PS}[3:0]}} ). See Section 19.3.2.</td>
</tr>
<tr>
<td>0000: ( \text{ck}_{ps} = \text{LCDCLK} )</td>
</tr>
<tr>
<td>0001: ( \text{ck}_{ps} = \frac{\text{LCDCLK}}{2} )</td>
</tr>
<tr>
<td>0002: ( \text{ck}_{ps} = \frac{\text{LCDCLK}}{4} )</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>1111: ( \text{ck}_{ps} = \frac{\text{LCDCLK}}{32768} )</td>
</tr>
<tr>
<td>Bits 21:18 <strong>DIV[3:0]</strong>: DIV clock divider</td>
</tr>
<tr>
<td>These bits are written by software to define the division factor of the DIV divider (see Section 19.3.2.)</td>
</tr>
<tr>
<td>0000: ( \text{ck}<em>{\text{div}} = \frac{\text{ck}</em>{ps}}{16} )</td>
</tr>
<tr>
<td>0001: ( \text{ck}<em>{\text{div}} = \frac{\text{ck}</em>{ps}}{17} )</td>
</tr>
<tr>
<td>0002: ( \text{ck}<em>{\text{div}} = \frac{\text{ck}</em>{ps}}{18} )</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>1111: ( \text{ck}<em>{\text{div}} = \frac{\text{ck}</em>{ps}}{31} )</td>
</tr>
</tbody>
</table>
Bits 17:16 **BLINK[1:0]**: Blink mode selection
- 00: Blink disabled
- 01: Blink enabled on SEG[0], COM[0] (1 pixel)
- 10: Blink enabled on SEG[0], all COMs (up to 8 pixels depending on the programmed duty)
- 11: Blink enabled on all SEGs and all COMs (all pixels)

Bits 15:13 **BLINKF[2:0]**: Blink frequency selection
- 000: $f_{\text{LCD}}/8$
- 001: $f_{\text{LCD}}/16$
- 010: $f_{\text{LCD}}/32$
- 011: $f_{\text{LCD}}/64$
- 100: $f_{\text{LCD}}/128$
- 101: $f_{\text{LCD}}/256$
- 110: $f_{\text{LCD}}/512$
- 111: $f_{\text{LCD}}/1024$

Bits 12:10 **CC[2:0]**: Contrast control
These bits specify one of the $V_{\text{LCD}}$ maximum voltages (independent of $V_{\text{DD}}$). It ranges from 2.60 V to 3.51V.
- 000: $V_{\text{LCD}0}$
- 001: $V_{\text{LCD}1}$
- 010: $V_{\text{LCD}2}$
- 011: $V_{\text{LCD}3}$
- 100: $V_{\text{LCD}4}$
- 101: $V_{\text{LCD}5}$
- 110: $V_{\text{LCD}6}$
- 111: $V_{\text{LCD}7}$

*Note: Refer to the datasheet for the $V_{\text{LCDx}}$ values.*

Bits 9:7 **DEAD[2:0]**: Dead time duration
These bits are written by software to configure the length of the dead time between frames. During the dead time the COM and SEG voltage levels are held at 0 V to reduce the contrast without modifying the frame rate.
- 000: No dead time
- 001: 1 phase period dead time
- 010: 2 phase period dead time
  ...
- 111: 7 phase period dead time
Bits 6:4 **PON[2:0]**: Pulse ON duration

These bits are written by software to define the pulse duration in terms of \( \text{ck}_\text{ps} \) pulses. A short pulse leads to lower power consumption, but displays with high internal resistance may need a longer pulse to achieve satisfactory contrast. Note that the pulse is never longer than one half prescaled LCD clock period.

- 000: 0
- 001: \( 1/\text{ck}_\text{ps} \)
- 010: \( 2/\text{ck}_\text{ps} \)
- 011: \( 3/\text{ck}_\text{ps} \)
- 100: \( 4/\text{ck}_\text{ps} \)
- 101: \( 5/\text{ck}_\text{ps} \)
- 110: \( 6/\text{ck}_\text{ps} \)
- 111: \( 7/\text{ck}_\text{ps} \)

PON duration example with LCDCLK = 32.768 kHz and PS=0x03:

- 000: 0 µs
- 001: 244 µs
- 010: 488 µs
- 011: 782 µs
- 100: 976 µs
- 101: 1.22 ms
- 110: 1.46 ms
- 111: 1.71 ms

Bit 3 **UDDIE**: Update display done interrupt enable

This bit is set and cleared by software.

- 0: LCD update display done interrupt disabled
- 1: LCD update display done interrupt enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SOFIE**: Start of frame interrupt enable

This bit is set and cleared by software.

- 0: LCD start-of-frame interrupt disabled
- 1: LCD start-of-frame interrupt enabled

Bit 0 **HD**: High drive enable

This bit is written by software to enable a low resistance divider. Displays with high internal resistance may need a longer drive time to achieve satisfactory contrast. This bit is useful in this case if some additional power consumption can be tolerated.

- 0: Permanent high drive disabled
- 1: Permanent high drive enabled. When HD = 1, PON[2:0] must be programmed to 001.

### 19.6.3 LCD status register (LCD_SR)

Address offset: 0x08

Reset value: 0x0000 0020

<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>
</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></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>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>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</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>rs</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>
Bits 31:6 Reserved, must be kept at reset value.

Bit 5  **FCRSF**: LCD frame control register synchronization flag

This bit is set by hardware each time the LCD_FCR register is updated in the LCDCLK domain. It is cleared by hardware when writing to the LCD_FCR register.

0: LCD frame control register not yet synchronized
1: LCD frame control register synchronized

Bit 4  **RDY**: Ready flag

This bit is set and cleared by hardware. It indicates the status of the stepup converter.

0: Not ready
1: Stepup converter enabled and ready to provide the correct voltage

Bit 3  **UDD**: Update display done

This bit is set by hardware. It is cleared by writing 1 to UDDC in LCD_CLR. The bit set has priority over the clear.

0: No event
1: Update display request done. A UDD interrupt is generated if UDDIE = 1 in LCD_FCR.

*Note: If the device is in Stop mode (PCLK not provided), UDD does not generate an interrupt even if UDDIE = 1. If the display is not enabled, the UDD interrupt never occurs.*

Bit 2  **UDR**: Update display request

Each time software modifies the LCD_RAM, it must set this bit to transfer the updated data to the second level buffer. This bit stays set until the end of the update. During this time, the LCD_RAM is write protected.

0: No effect
1: Update display request

*Note: When the display is disabled, the update is performed for all LCD_DISPLAY locations. When the display is enabled, the update is performed only for locations for which commons are active (depending on DUTY). For example if DUTY = 1/2, only the LCD_DISPLAY of COM0 and COM1 are updated.

Writing 0 on this bit or writing 1 when it is already 1 has no effect. This bit can be cleared by hardware only. It can be cleared only when LCDEN = 1*

Bit 1  **SOF**: Start-of-frame flag

This bit is set by hardware at the beginning of a new frame, at the same time as the display data is updated. It is cleared by writing a 1 to SOFC in LCD_CLR. The bit clear has priority over the set.

0: No event
1: Start-of-frame event occurred. An LCD SOF interrupt is generated if SOFIE is set.

Bit 0  **ENS**: LCD enabled status

This bit is set and cleared by hardware. It indicates the LCD controller status.

0: LCD controller disabled
1: LCD controller enabled

*Note: This bit is set immediately when LCDEN in LCD_CR goes from 0 to 1. On deactivation, it reflects the real LCD status. It becomes 0 at the end of the last displayed frame.*
19.6.4 LCD clear register (LCD_CLR)

Address offset: 0x0C
Reset value: 0x0000 0000

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

Bit 3 **UDDC**: Update display done clear
   This bit is written by software to clear UDD in LCD_SR.
   0: No effect
   1: Clear UDD flag.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SOFC**: Start-of-frame flag clear
   This bit is written by software to clear SOF in LCD_SR.
   0: No effect
   1: Clear SOF flag.

Bit 0 Reserved, must be kept at reset value.

19.6.5 LCD display memory (LCD_RAMx)

Address offset: 0x14 + 0x4 * x, (x = 0, 2, 4, 6, 8, 10, 12, 14)
Reset value: 0x0000 0000

Bits 31:0 **SEGMENT_DATA[31:0]**:
Each bit corresponds to one pixel of the LCD display.
0: Pixel inactive
1: Pixel active
19.6.6 LCD display memory (LCD_RAMx)

Address offset: 0x14 + 0x4 * x, (x = 1, 3, 5, 7)
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>

SEGMENT_DATA[43:32]

- Bits 31:12 Reserved, must be kept at reset value.
- Bits 11:0 SEGMENT_DATA[43:32]:
  - Each bit corresponds to one pixel of the LCD display.
  - 0: Pixel inactive
  - 1: Pixel active

19.6.7 LCD display memory (LCD_RAMx)

Address offset: 0x14 + 0x4 * x, (x = 9, 11, 13, 15)
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>

SEGMENT_DATA[39:32]

- Bits 31:8 Reserved, must be kept at reset value.
- Bits 7:0 SEGMENT_DATA[39:32]:
  - Each bit corresponds to one pixel of the LCD display.
  - 0: Pixel inactive
  - 1: Pixel active

19.6.8 LCD register map

Table 105. LCD 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  |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

- 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
| 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 |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x04   | LCD_FCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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   | LCD_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x0C   | LCD_CLR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | LCD_RAM (COM0) | S31 | S30 | S29 | S28 | S27 | S26 | S25 | S24 | S23 | S22 | S21 | S20 | S19 | S18 | S17 | S16 | S15 | S14 | S13 | S12 | S11 | S10 | S9  | S8  | S7  | S6  | S5  | S4  | S3  | S2  | S1  | S0  |
|        | 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   | LCD_RAM (COM1) | S31 | S30 | S29 | S28 | S27 | S26 | S25 | S24 | S23 | S22 | S21 | S20 | S19 | S18 | S17 | S16 | S15 | S14 | S13 | S12 | S11 | S10 | S9  | S8  | S7  | S6  | S5  | S4  | S3  | S2  | S1  | S0  |
|        | 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  |
| 0x24   | LCD_RAM (COM2) | S31 | S30 | S29 | S28 | S27 | S26 | S25 | S24 | S23 | S22 | S21 | S20 | S19 | S18 | S17 | S16 | S15 | S14 | S13 | S12 | S11 | S10 | S9  | S8  | S7  | S6  | S5  | S4  | S3  | S2  | S1  | S0  |
|        | 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  |
| 0x28   | LCD_RAM (COM3) | S31 | S30 | S29 | S28 | S27 | S26 | S25 | S24 | S23 | S22 | S21 | S20 | S19 | S18 | S17 | S16 | S15 | S14 | S13 | S12 | S11 | S10 | S9  | S8  | S7  | S6  | S5  | S4  | S3  | S2  | S1  | S0  |
|        | 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  |
| 0x34   | LCD_RAM (COM4) | S31 | S30 | S29 | S28 | S27 | S26 | S25 | S24 | S23 | S22 | S21 | S20 | S19 | S18 | S17 | S16 | S15 | S14 | S13 | S12 | S11 | S10 | S9  | S8  | S7  | S6  | S5  | S4  | S3  | S2  | S1  | S0  |
|        | 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 105. LCD register map and reset values (continued)
### Table 105. LCD 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 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x3C   | LCD_RAM (COM5) | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x40   |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x44   | LCD_RAM (COM6) | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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   |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4C   | LCD_RAM (COM7) | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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   |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.2 for the register boundary addresses table.
20 Touch sensing controller (TSC)

This section applies to STM32WB55xx devices only.

20.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode that is protected from direct touch by a dielectric (for example glass, plastic). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMTouch touch sensing firmware library, which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

20.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 28 capacitive sensing channels without shield
- Supports up to 6 capacitive sensing channels when shield is used in group 7 (recommended mode)
- Spread spectrum feature to improve system robustness in noisy environments
- Full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.
20.3 TSC functional description

20.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in Figure 108.

![Figure 108. TSC block diagram](MS30929V1)

20.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group composed of up to four GPIOs (see Figure 109). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor $C_S$. Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.
The surface charge transfer acquisition principle consists of charging an electrode capacitance (C_X) and transferring a part of the accumulated charge into a sampling capacitor (C_S). This sequence is repeated until the voltage across C_S reaches a given threshold (V_{IH} in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

*Table 106* details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across C_S reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor R_S improves the ESD immunity of the solution.
Table 106. Acquisition sequence summary

<table>
<thead>
<tr>
<th>State</th>
<th>Gx_IO1 (channel)</th>
<th>Gx_IO2 (sampling)</th>
<th>Gx_IO3 (channel)</th>
<th>Gx_IO4 (channel)</th>
<th>State description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Input floating with analog switch closed</td>
<td>Output open-drain low with analog switch closed</td>
<td>Input floating with analog switch closed</td>
<td></td>
<td>Discharge all $C_X$ and $C_S$</td>
</tr>
<tr>
<td>#2</td>
<td>Input floating</td>
<td></td>
<td></td>
<td></td>
<td>Dead time</td>
</tr>
<tr>
<td>#3</td>
<td>Output push-pull high</td>
<td>Input floating</td>
<td></td>
<td></td>
<td>Charge $C_{X1}$</td>
</tr>
<tr>
<td>#4</td>
<td>Input floating</td>
<td></td>
<td></td>
<td></td>
<td>Dead time</td>
</tr>
<tr>
<td>#5</td>
<td>Input floating with analog switch closed</td>
<td>Input floating</td>
<td></td>
<td></td>
<td>Charge transfer from $C_{X1}$ to $C_S$</td>
</tr>
<tr>
<td>#6</td>
<td>Input floating</td>
<td></td>
<td></td>
<td></td>
<td>Dead time</td>
</tr>
<tr>
<td>#7</td>
<td>Input floating</td>
<td></td>
<td></td>
<td></td>
<td>Measure $C_S$ voltage</td>
</tr>
</tbody>
</table>

Note: $Gx_{IOy}$ where $x$ is the analog I/O group number and $y$ the GPIO number within the selected group.

The voltage variation over the time on the sampling capacitor $C_S$ is detailed below (refer to Figure 109 for $V_{SENSOR}$ and $V_{CS}$ definition):

Figure 110. Sampling capacitor voltage variation
20.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register.
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register.

The reset and clock controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, refer to Section 8: Reset and clock control (RCC).

20.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in Figure 111.

For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of CX) and the pulse low state (transfer of charge from CX to C₃) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 µs. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that CX is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 1 period of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.
The reading of the sampling capacitor I/O, to determine if the voltage across $C_S$ has reached the given threshold, is performed at the end of the pulse low state.

**Note:** The following TSC control register configurations are forbidden:
- bits $PGPSC$ are set to ‘000’ and bits $CTPL$ are set to ‘0001’
- bits $PGPSC$ are set to ‘001’ and bits $CTPL$ are set to ‘0000’
20.3.5 Spread spectrum feature

The spread spectrum feature generates a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10% to 50% of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 kHz (4 µs), the typical spread spectrum deviation is 10% (400 ns) which leads to a minimum charge transfer frequency of ~227 kHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

![Figure 112. Spread spectrum variation principle](image)

The table below details the maximum frequency deviation with different HCLK settings:

<table>
<thead>
<tr>
<th>f_{HCLK}</th>
<th>Spread spectrum step</th>
<th>Maximum spread spectrum deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 MHz</td>
<td>31.25 ns</td>
<td>8206 ns</td>
</tr>
</tbody>
</table>

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC_CR register.

20.3.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC_CR register. As soon as an acquisition group counter reaches this maximum value, the ongoing acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.
20.3.7 **Sampling capacitor I/O and channel I/O mode selection**

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC_IOSCR and TSC_IOCCR register.

When there is no ongoing acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is ongoing, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

<table>
<thead>
<tr>
<th>IODEF bit</th>
<th>Acquisition status</th>
<th>Unused I/O mode</th>
<th>Channel I/O mode</th>
<th>Sampling capacitor I/O mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No</td>
<td>Output push-pull low</td>
<td>Output push-pull low</td>
<td>Output push-pull low</td>
</tr>
<tr>
<td>0</td>
<td>Ongoing</td>
<td>Output push-pull low</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>No</td>
<td>Input floating</td>
<td>Input floating</td>
<td>Input floating</td>
</tr>
<tr>
<td>1</td>
<td>Ongoing</td>
<td>Input floating</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx_IOy bit in the TSC_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx_IOy bit in the TSC_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

**Note:** *During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR or TSC_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*
20.3.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC_IOGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The C_S voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they are pulsed.

When the C_S voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC_IOGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC_ISR register is set. An interrupt request is generated if the EOAIE bit in the TSC_IER register is set.

In the case that a max count error is detected, the ongoing acquisition is stopped and both the EOAF and MCEF flags in the TSC_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAIE and MCEIE bits of the TSCIER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAIC and MCEIC bits in the TSC_ICR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

20.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller is able to take the control of the Schmitt trigger hysteresis and analog switch of each Gx_IOy. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx_IOy bit in the TSC_IOHCR register.
20.4 TSC low-power modes

Table 109. Effect of low-power modes on TSC

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. Peripheral 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. Peripheral interrupts cause the device to exit Low-power sleep mode.</td>
</tr>
<tr>
<td>Stop 0 / Stop 1</td>
<td>Peripheral registers content is kept.</td>
</tr>
<tr>
<td>Stop 2</td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>Powered-down. The peripheral must be reinitialized after exiting Standby or Shutdown mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td></td>
</tr>
</tbody>
</table>

20.5 TSC interrupts

Table 110. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Enable control bit</th>
<th>Event flag</th>
<th>Clear flag</th>
<th>Exit the Sleep mode</th>
<th>Exit the Stop mode</th>
<th>Exit the Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>End of acquisition</td>
<td>EOAIE</td>
<td>EOAIF</td>
<td>EOAIC</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Max count error</td>
<td>MCEIE</td>
<td>MCEIF</td>
<td>MCEIC</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

20.6 TSC registers

Refer to Section 1.2 on page 61 of the reference manual for a list of abbreviations used in register descriptions.

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

20.6.1 TSC control register (TSC_CR)

Address offset: 0x00
Reset value: 0x0000 0000
Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of $C_X$).

- 0000: $1 \times t_{PGCLK}$
- 0001: $2 \times t_{PGCLK}$
- ... 
- 1111: $16 \times t_{PGCLK}$

*Note:* These bits must not be modified when an acquisition is ongoing.

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from $C_X$ to $C_S$).

- 0000: $1 \times t_{PGCLK}$
- 0001: $2 \times t_{PGCLK}$
- ... 
- 1111: $16 \times t_{PGCLK}$

*Note:* These bits must not be modified when an acquisition is ongoing.

*Note:* Some configurations are forbidden. Refer to the Section 20.3.4: Charge transfer acquisition sequence for details.

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

- 0000000: $1 \times t_{SSCLK}$
- 0000001: $2 \times t_{SSCLK}$
- ... 
- 1111111: $128 \times t_{SSCLK}$

*Note:* These bits must not be modified when an acquisition is ongoing.

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

- 0: Spread spectrum disabled
- 1: Spread spectrum enabled

*Note:* This bit must not be modified when an acquisition is ongoing.

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

- 0: $f_{HCLK}$
- 1: $f_{HCLK}/2$

*Note:* This bit must not be modified when an acquisition is ongoing.
Bits 14:12 **PGPSC[2:0]**: Pulse generator prescaler
These bits are set and cleared by software. They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

- 000: fHCLK
- 001: fHCLK /2
- 010: fHCLK /4
- 011: fHCLK /8
- 100: fHCLK /16
- 101: fHCLK /32
- 110: fHCLK /64
- 111: fHCLK /128

*Note:* These bits must not be modified when an acquisition is ongoing.

*Note:* Some configurations are forbidden. Refer to the Section 20.3.4: Charge transfer acquisition sequence for details.

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value
These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

- 000: 255
- 001: 511
- 010: 1023
- 011: 2047
- 100: 4095
- 101: 8191
- 110: 16383
- 111: reserved

*Note:* These bits must not be modified when an acquisition is ongoing.

Bit 4 **IODEF**: I/O Default mode
This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no ongoing acquisition. When there is an ongoing acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

- 0: I/Os are forced to output push-pull low
- 1: I/Os are in input floating

*Note:* This bit must not be modified when an acquisition is ongoing.

Bit 3 **SYNCPOL**: Synchronization pin polarity
This bit is set and cleared by software to select the polarity of the synchronization input pin.

- 0: Falling edge only
- 1: Rising edge and high level
20.6.2 **TSC interrupt enable register (TSC_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>
<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, must be kept at reset value.

**Bit 1 MCEIE:** Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

0: Max count error interrupt disabled
1: Max count error interrupt enabled

**Bit 0 EOAIE:** End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

0: End of acquisition interrupt disabled
1: End of acquisition interrupt enabled
20.6.3 TSC interrupt clear register (TSC_ICR)

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>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, must be kept at reset value.

Bit 1 **MCEIC**: Max count error interrupt clear
This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.
0: No effect
1: Clears the corresponding MCEF of the TSC_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear
This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.
0: No effect
1: Clears the corresponding EOAF of the TSC_ISR register

20.6.4 TSC interrupt status register (TSC_ISR)

Address offset: 0x0C
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, must be kept at reset value.

Bit 1 **MCEF**: Max count error flag
This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC_ICR register.
0: No max count error (MCE) detected
1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag
This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAIC of the TSC_ICR register.
0: Acquisition is ongoing or not started
1: Acquisition is complete
20.6.5 TSC I/O hysteresis control register (TSC_IOHCR)

Address offset: 0x10
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>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>G7_IO4</td>
<td>G7_IO3</td>
<td>G7_IO2</td>
<td>G7_IO1</td>
<td>G6_IO4</td>
<td>G6_IO3</td>
<td>G6_IO2</td>
<td>G6_IO1</td>
<td>G5_IO4</td>
<td>G5_IO3</td>
<td>G5_IO2</td>
<td>G5_IO1</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 31:28 Reserved, must be kept at reset value.

Bits 27:0 **Gx_IOy**: Gx_I0y Schmitt trigger hysteresis mode
These bits are set and cleared by software to enable/disable the Gx_IOy Schmitt trigger hysteresis.
0: Gx_IOy Schmitt trigger hysteresis disabled
1: Gx_IOy Schmitt trigger hysteresis enabled

*Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).*

20.6.6 TSC I/O analog switch control register (TSC_IOASCR)

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>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>G7_IO4</td>
<td>G7_IO3</td>
<td>G7_IO2</td>
<td>G7_IO1</td>
<td>G6_IO4</td>
<td>G6_IO3</td>
<td>G6_IO2</td>
<td>G6_IO1</td>
<td>G5_IO4</td>
<td>G5_IO3</td>
<td>G5_IO2</td>
<td>G5_IO1</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 31:28 Reserved, must be kept at reset value.

Bits 27:0 **Gx_IOy**: Gx_I0y analog switch enable
These bits are set and cleared by software to enable/disable the Gx_IOy analog switch.
0: Gx_IOy analog switch disabled (opened)
1: Gx_IOy analog switch enabled (closed)

*Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).*
### 20.6.7 TSC I/O sampling control register (TSC_IOSCR)

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>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>G7_IO4</td>
<td>G7_IO3</td>
<td>G7_IO2</td>
<td>G7_IO1</td>
<td>G6_IO4</td>
<td>G6_IO3</td>
<td>G6_IO2</td>
<td>G6_IO1</td>
<td>G5_IO4</td>
<td>G5_IO3</td>
<td>G5_IO2</td>
<td>G5_IO1</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>
<tr>
<td>G4_IO4</td>
<td>G4_IO3</td>
<td>G4_IO2</td>
<td>G4_IO1</td>
<td>G3_IO4</td>
<td>G3_IO3</td>
<td>G3_IO2</td>
<td>G3_IO1</td>
<td>G2_IO4</td>
<td>G2_IO3</td>
<td>G2_IO2</td>
<td>G2_IO1</td>
<td>G1_IO4</td>
<td>G1_IO3</td>
<td>G1_IO2</td>
<td>G1_IO1</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 31:28 Reserved, must be kept at reset value.  
Bits 27:0 **Gx_IOy**: Gx_IOy sampling mode  
These bits are set and cleared by software to configure the Gx_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.  
0: Gx_IOy unused  
1: Gx_IOy used as sampling capacitor  
**Note**: These bits must not be modified when an acquisition is ongoing. Di 
During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

### 20.6.8 TSC I/O channel control register (TSC_IOCCCR)

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>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>G7_IO4</td>
<td>G7_IO3</td>
<td>G7_IO2</td>
<td>G7_IO1</td>
<td>G6_IO4</td>
<td>G6_IO3</td>
<td>G6_IO2</td>
<td>G6_IO1</td>
<td>G5_IO4</td>
<td>G5_IO3</td>
<td>G5_IO2</td>
<td>G5_IO1</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>
<tr>
<td>G4_IO4</td>
<td>G4_IO3</td>
<td>G4_IO2</td>
<td>G4_IO1</td>
<td>G3_IO4</td>
<td>G3_IO3</td>
<td>G3_IO2</td>
<td>G3_IO1</td>
<td>G2_IO4</td>
<td>G2_IO3</td>
<td>G2_IO2</td>
<td>G2_IO1</td>
<td>G1_IO4</td>
<td>G1_IO3</td>
<td>G1_IO2</td>
<td>G1_IO1</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 31:28 Reserved, must be kept at reset value.  
Bits 27:0 **Gx_IOy**: Gx_IOy channel mode  
These bits are set and cleared by software to configure the Gx_IOy as a channel I/O.  
0: Gx_IOy unused  
1: Gx_IOy used as channel  
**Note**: These bits must not be modified when an acquisition is ongoing.  
During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC_IOCCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.
20.6.9 TSC I/O group control status register (TSC_IOGCSR)

Address offset: 0x30
Reset value: 0x0000 0000

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **GxS**: Analog I/O group x status
These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.

0: Acquisition on analog I/O group x is ongoing or not started
1: Acquisition on analog I/O group x is complete

*Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.*

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **GxE**: Analog I/O group x enable
These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.

0: Acquisition on analog I/O group x disabled
1: Acquisition on analog I/O group x enabled

20.6.10 TSC I/O group x counter register (TSC_IOGxCR)

x represents the analog I/O group number.

Address offset: 0x30 + 0x04 * x, (x = 1 to 7)
Reset value: 0x0000 0000

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **CNT[13:0]**: Counter value
These bits represent the number of charge transfer cycles generated on the analog I/O group x to complete its acquisition (voltage across Cs has reached the threshold).
## 20.6.11 TSC register map

### Table 111. TSC register map and reset values

<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>0x0000</td>
<td>TSC_CR</td>
<td>Reset</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>0x0004</td>
<td>TSC_IER</td>
<td>Reset</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>0x0008</td>
<td>TSC_ICR</td>
<td>Reset</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>0x000C</td>
<td>TSC_ISR</td>
<td>Reset</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>0x0010</td>
<td>TSC_IOHCR</td>
<td>Reset</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>0x0014</td>
<td>TSC_IOASCR</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>
</tr>
<tr>
<td>0x0018</td>
<td>TSC_IOASCR</td>
<td>Reset</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>0x001C</td>
<td>TSC_IOASCR</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>
</tr>
<tr>
<td>0x0020</td>
<td>TSC_IOASCR</td>
<td>Reset</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>0x0024</td>
<td>TSC_IOASCR</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>
</tr>
<tr>
<td>0x0028</td>
<td>TSC_IOASCR</td>
<td>Reset</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>0x002C</td>
<td>TSC_IOASCR</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>
</tr>
<tr>
<td>0x0030</td>
<td>TSC_IOGCSR</td>
<td>Reset</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>0x0034</td>
<td>TSC_IOG1CR</td>
<td>Reset</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>0x0038</td>
<td>TSC_IOG2CR</td>
<td>Reset</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 on page 66 for the register boundary addresses.
21 True random number generator (RNG)

21.1 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 the German BSI statistical tests of AIS-31 (T0 to T8).

21.2 RNG main features

- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- In the NIST configuration, 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).
21.3 RNG functional description

21.3.1 RNG block diagram

Figure 113 shows the RNG block diagram.

Figure 113. RNG block diagram

21.3.2 RNG internal signals

Table 112 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 112. RNG internal input/output signals

<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>
21.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. Within its boundary the RNG implements the entropy source model pictured on *Figure 114*.

It includes an analog noise source, a digitization stage with post-processing, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

*Figure 114. Entropy source model*

The components pictured above are detailed hereafter.

**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 21.3.8: RNG low-power use*.

- 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 (rng_hclk).

*Note:* In *Section 21.6: RNG entropy source validation* the 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 the 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).

Also note that post-processing computations are triggered when at least 32 bits of raw datum is 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 21.5: 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 changes from 0 to 1. This flag remains high until the 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.

1. Continuous health tests, running indefinitely on the output of the noise source
   - Repetition count test, flagging an error when:
     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
   - 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 21.3.7: Error management.

Note: An interrupt can be generated when an error is detected.

21.3.4 RNG initialization

The RNG simplified state machine is pictured on Figure 115.

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 the 128-bit conditioning shift register.
2. The conditioning logic is enabled and the 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 the post processing result.
4. The output buffer is refilled automatically according to the RNG usage.
The associated initialization time can be found in Section 21.5: RNG processing time.

Figure 115. RNG initialization overview

21.3.5 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 must 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 the 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 is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in Section 21.3.7 must 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 must be set to 0)
   – A random number is ready (the DRDY bit must 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 is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in Section 21.3.7 must be used.

Note: When data is not ready (DRDY = 0) RNG_DR returns zero. It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in Section 21.3.8: RNG low-power use.

Software post-processing

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 21.3.3: Random number generation.

21.3.6 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 21.6: RNG entropy source validation.

Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency the must be higher than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).

See Section 21.3.1: RNG block diagram for details (AHB and RNG clock domains).

21.3.7 Error management

In parallel to random number generation a 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 must check that the RNG clock is configured correctly (see
Section 21.3.6: RNG clocking) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

**Note:** The clock error has no impact on generated random numbers that is the application can still read the 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 is automatically restarted by the RNG.

The following sequence must 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.

#### 21.3.8 RNG low-power use

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 the 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 are collected from the noise source and a complete conditioning round is 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 (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1.
21.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:
- Data ready flag
- Seed error, see Section 21.3.7: Error management
- Clock error, see Section 21.3.7: Error management

Dedicated interrupt enable control bits are available as shown in Table 113.

<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</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>

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.

21.5 RNG processing time

The conditioning stage can produce four 32-bit random numbers every $16 \times \frac{f_{AHB}}{f_{RNG}}$ 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 21.3.4: RNG initialization). Indeed, after enabling the RNG for the first time, random data is first available after either:

- $128 \times f_{RNG}$ clock cycles + $426 \times f_{AHB}$ cycles, if $f_{AHB} < f_{\text{threshold}}$
- $192 \times f_{RNG}$ clock cycles + $213 \times f_{AHB}$ cycles, if $f_{AHB} \geq f_{\text{threshold}}$

With $f_{\text{threshold}} = \left(213 \times f_{RNG}\right)/64$

21.6 RNG entropy source validation

21.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8).

21.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz (CED bit = ‘0’ in RNG_CR register) and rng_clk = 400 kHz (CED bit = ‘1’ in RNG_CR register).
21.7 **RNG registers**

The RNG is associated with a control register, a data register and a status register.

### 21.7.1 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>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></td>
<td></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 enabled
- 1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is 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 the DRDY, SEIS, or CEIS is set 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.

### 21.7.2 RNG status register (RNG_SR)

**Address offset: 0x004**

**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>rc_w0</td>
<td>rc_w0</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>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.
21.7.3 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 the DRDY = 1 and the value is not 0x0, 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, the RNDATA value is zero.
When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

### 21.7.4 RNG register 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  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | RNG_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  |
| 0x008  | RNG_DR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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: Memory organization* for the register boundary addresses.
22 AES hardware accelerator (AES)

22.1 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.

The peripheral supports CTR, GCM, GMAC, CCM, ECB, and CBC chaining modes for key sizes of 128 or 256 bits.

AES is an AMBA AHB slave peripheral accessible through 32-bit single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels required).

22.2 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 round key scheduler to compute the last round key for ECB/CBC decryption
- 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

22.3 AES implementation

The devices have two AES peripherals, AES1 and AES2.
22.4 AES functional description

22.4.1 AES block diagram

*Figure 116* shows the block diagram of AES.

![Figure 116. AES block diagram](image)

22.4.2 AES internal signals

*Table 115* 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>Input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>aes_it</td>
<td>Output</td>
<td>AES interrupt request</td>
</tr>
<tr>
<td>aes_in_dma</td>
<td>Input/Output</td>
<td>Input DMA single request/acknowledge</td>
</tr>
<tr>
<td>aes_out_dma</td>
<td>Input/Output</td>
<td>Output DMA single request/acknowledge</td>
</tr>
</tbody>
</table>

22.4.3 AES cryptographic core

**Overview**

The AES cryptographic core consists of the following components:
- AES core 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 128-bit initialization vector IV.

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 22.4.4: AES procedure to perform a cipher operation on page 596.

**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 22.4.8: AES basic chaining modes (ECB, CBC).
Electronic codebook (ECB) mode

Figure 117. 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 118. 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 XOR-ed 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 use is not recommended by NIST.

**22.4.4 AES procedure to perform a cipher operation**

**Introduction**

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 22.4.8: AES basic chaining modes (ECB, CBC)*.
**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, select Mode 1 (MODE[1:0] = 00).
  - For decryption, select Mode 3 (MODE[1:0] = 10), unless ECB or CBC chaining modes are used. In this latter case, perform an initial key derivation of the encryption key, as described in Section 22.4.5: AES decryption round key preparation.

- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the AES_CR register.

- 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.

- 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).

**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 22.4.6: 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_DOUR 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. As it is the last block, discard the data that is not part of the data, 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 a 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 data, 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 22.4.16: 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 of GCM, GMAC, and CCM chaining modes.
22.4.5 AES decryption round key preparation

Internal key schedule is used to generate AES round keys. In AES encryption, the round 0 key is the one stored in the key registers. AES decryption must start using the last round key. As the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This key scheduling is only required for AES decryption in ECB and CBC modes.

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 123. 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 123 (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 123. 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).
22.4.6 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 22.4.4: AES procedure to perform a cipher operation on page 596 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 22.4.13: AES data registers and data swapping for details).

22.4.7 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 124* gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

*Figure 124. Example of suspend mode management*
A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

22.4.8 AES basic chaining modes (ECB, CBC)

Overview

This section gives a brief explanation of the four basic operation modes provided by the AES core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.

*Figure 125* illustrates the electronic codebook (ECB) encryption.

*Figure 125. 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 126* illustrates the electronic codebook (ECB) decryption.

*Figure 126. 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\textsubscript{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\textsubscript{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 127* illustrates the cipher block chaining (CBC) encryption.

In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping ($P_1'$), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the $I_1$ input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block $O_1$, after swapping operation, is used as ciphertext $C_1$. The $O_1$ data is then XOR-ed with the second-block plaintext data $P_2'$ to produce the $I_2$ 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 22.4.6: AES ciphertext stealing and data padding*. *Figure 128* 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 22.4.6: AES ciphertext stealing and data padding.

For more information on data swapping, refer to Section 22.4.13: AES data registers and data swapping.

**ECB/CBC encryption sequence**

The sequence of events to perform an ECB/CBC encryption (more detail in Section 22.4.4):

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 129.
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 129. 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 129. ECB/CBC encryption (Mode 1)**
**ECB/CBC decryption sequence**

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 22.4.4*).

1. Follow the steps described in *Section 22.4.5: AES decryption round key preparation*, 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 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield. KEYSIZE bitfield must be kept as-is.
3. Write the AES_IVRx registers with the initialization vector (required in CBC mode only).
4. Enable AES by setting the EN bit of the AES_CR register.
5. Write the AES_DINR register four times to input the cipher text (MSB first), as shown in *Figure 130*.
6. Wait until the CCF flag is set in the AES_SR register.
7. Read the AES_DOUTR register four times to get the plain text (MSB first), as shown in *Figure 130*. Then clear the CCF flag by setting the CCFC bit of the AES_CR register.
8. Repeat steps 5-6-7 to process all the blocks encrypted with the same key.

*Figure 130. 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 AES_CR register and 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. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Restore AES_CR register (with correct KEYSIZE) then restore AES_KEYRx registers. 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 22.4.5: AES decryption round key preparation* (only required for ECB or CBC decryption). This step is not necessary if derived key information is 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 0x0 or 0x1, 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.
22.4.9 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 131.

Figure 131. Message construction in CTR mode

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 must 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 132 and Figure 133 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.
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 116.

**Table 116. 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>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>32-bit counter = 0x0001</td>
</tr>
</tbody>
</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 setting 01 (key derivation) is forbidden and all the other settings default to encryption mode.

The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
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 116.
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 22.4.4: 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 22.4.8: AES basic chaining modes (ECB, CBC).

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

22.4.10 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 134.
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 a 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 117.

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 117. GCM last block definition

<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>Input data</td>
<td>0x0</td>
<td>AAD length[31:0]</td>
<td>0x0</td>
<td>Payload length[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**GCM processing**

Figure 135 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 135. 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 118).

Table 118. Initialization of AES_IVRx registers in GCM mode

<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>
<tbody>
<tr>
<td>32-bit counter = 0x0002</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend**

- XOR
- Input
- Output

**Note:** In this 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 \textit{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. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
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 \textit{Table 118}.
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 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 \textit{Section 22.4.4: AES procedure to perform a cipher operation}. No data is read during this phase.
4. Repeat the step 3 until the last additional authenticated data block is processed.

\textit{Note: The header phase can be skipped if there is no AAD, that is, Len(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 22.4.4: AES procedure to perform a cipher operation on page 596, 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 117. 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 of the AES_SR register, by setting 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 is written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

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 the CCFC bit of the AES CR register.
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. Disable the AES peripheral by clearing the EN bit of the AES CR register.
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.

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.

22.4.11 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 136.

**Figure 136. Message construction in GMAC mode**

![Diagram showing message construction in GMAC mode](MSv42158V2)

**AES GMAC processing**

Figure 137 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 137. GMAC authentication mode**

![Diagram showing GMAC authentication mode](MSv42150V2)

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.

### 22.4.12 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 138.

**Figure 138. 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 must 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 must 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 $\text{Len}(A)$ that can be a non-multiple of 16 bytes (see Figure 138). 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 $\text{Len}(P)$. This length can be a non-multiple of 16 bytes (see Figure 138).

**Encrypted MAC (T)** of length $\text{Len}(T)$ appended to the ciphertext C of overall length $\text{Len}(C)$.

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: 10111213 141516 (Len(N) = 56 bits or 7 bytes)
- A: 00010203 04050607 (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 00000000 00000000
- CTR0: 0710111213 141516 00000000 00000000
- CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.
CCM processing

*Figure 139* 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 119* 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 119. Initialization of AES_IVRx registers in CCM mode**

<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>
<tbody>
<tr>
<td>B0[127:96]</td>
<td>B0[95:64]</td>
<td>B0[63:32]</td>
<td>B0[31:0]</td>
</tr>
</tbody>
</table>
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. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
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 119.
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. 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 22.4.4: AES procedure to perform a cipher operation. No data is read during this phase.
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 22.4.4: AES procedure to perform a cipher operation on page 596, 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, 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(C) = 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, swapping is applied to tag data read from AES_DOUTR register.

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. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
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.

22.4.13 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[31:0]), 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[31:0]), 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 22.4.16: 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 140 shows the construction of AES processing core input buffer data P127 to P0, 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 to P0.

**Figure 140. 128-bit block construction with respect to data swap**

<table>
<thead>
<tr>
<th>DATATYPE[1:0]</th>
<th>00: no swapping</th>
<th>01: 16-bit (half-word) swapping</th>
<th>10: 8-bit (byte) swapping</th>
<th>11: bit swapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td>Word 3</td>
<td>Word 2</td>
<td>Word 1</td>
<td>Word 0</td>
</tr>
<tr>
<td>D127</td>
<td>D96</td>
<td>D64</td>
<td>D32</td>
<td>D0</td>
</tr>
<tr>
<td></td>
<td>D95</td>
<td>D63</td>
<td>D31</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D62</td>
<td>D30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D61</td>
<td>D29</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D60</td>
<td>D28</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D59</td>
<td>D27</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D58</td>
<td>D25</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D57</td>
<td>D23</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D56</td>
<td>D21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D55</td>
<td>D19</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D54</td>
<td>D17</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D53</td>
<td>D15</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D52</td>
<td>D13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D51</td>
<td>D11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D50</td>
<td>D9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D49</td>
<td>D7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D48</td>
<td>D5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D47</td>
<td>D3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D46</td>
<td>D1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D44</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D43</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D42</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D41</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D40</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D39</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D38</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D37</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D36</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D35</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D34</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D33</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D32</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D31</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D30</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D29</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D27</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D26</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D25</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D24</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D23</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D22</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D21</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D20</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D19</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D17</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D15</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D14</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D13</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- AES input/output data block in memory
- AES core input/output buffer data
- Zero padding (example)
- Data swap
- MSB: most significant bit (127) of memory data block / AES core buffer
- LSB: least significant bit (0) of memory data block / AES core buffer
- Order of write to AES_DINR / read from AES_DOUTR input/output data bit 'x'

MSv42153V2
Note: The data in AES key registers (AES_KEYRx) and initialization registers (AES_IVRx) are not sensitive to the swap mode selection.

Data padding

Figure 140 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

22.4.14 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 eight registers as shown in Table 120.

Table 120. 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>
</table>

The key for encryption or decryption may be written into these registers when the AES peripheral is disabled, by clearing the EN bit of the AES_CR register.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the AES_CR register.

22.4.15 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.
22.4.16 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 141.

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. For details, refer to Section 22.4.4: AES procedure to perform a cipher operation.

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 142.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.
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 has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to data transferring managed by software. See Section 22.4.4: AES procedure to perform a cipher operation, subsection Data append, for details.

22.4.17 AES error management

AES configuration can be changed at any moment by clearing the EN bit of the AES_CR register.

Read error flag (RDERR)

Unexpected read attempt of the AES_DOUTR register sets the RDERR flag of the AES_SR register, and returns zero.

RDERR is triggered during the computation phase or during the input phase.

Note: AES is not disabled upon a RDERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to Section 22.5: AES interrupts.

The RDERR flag is cleared by setting the ERRIE bit of the AES_CR register.

Write error flag (WDERR)

Unexpected write attempt of the AES_DINR register sets the WRERR flag of the AES_SR register, and has no effect on the AES_DINR register. The WRERR is triggered during the computation phase or during the output phase.
**Note:** AES is not disabled after a WRERR error detection and continues processing. An interrupt is generated if the ERRIE bit of the AES_CR register is set. For more details, refer to Section 22.5: AES interrupts.

The WRERR flag is cleared by setting the ERRC bit of the AES_CR register.

### 22.5 AES interrupts

Individual maskable interrupt sources generated by the AES peripheral signal the following events:
- computation completed
- read error
- write error

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Arm® Cortex® interrupt controller. Each can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the AES_CR register, and cleared by setting the corresponding bit of the AES_CR register.

The status of each can be read from the AES_SR register. *Table 121* gives a summary of the interrupt sources, their event flags and enable bits.

<table>
<thead>
<tr>
<th>AES interrupt event</th>
<th>Event flag</th>
<th>Enable bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>computation completed flag</td>
<td>CCF</td>
<td>CCFIE</td>
<td>set CCFC(1)</td>
</tr>
<tr>
<td>read error flag</td>
<td>RDERR</td>
<td>ERRIE</td>
<td>set ERRRC(1)</td>
</tr>
<tr>
<td>write error flag</td>
<td>WRERR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Bit of the AES_CR register.

### 22.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Algorithm</th>
<th>Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Mode 1: Encryption</td>
<td>ECB, CBC, CTR</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Mode 2: Key derivation</td>
<td>-</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>Mode 3: Decryption</td>
<td>ECB, CBC, CTR</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Mode 4: Key derivation then decryption</td>
<td>ECB, CBC</td>
<td>106</td>
</tr>
</tbody>
</table>
Table 122. Processing latency for ECB, CBC and CTR (continued)

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Algorithm</th>
<th>Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>256-bit</td>
<td>Mode 1: Encryption</td>
<td>ECB, CBC, CTR</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Mode 2: Key derivation</td>
<td>-</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>Mode 3: Decryption</td>
<td>ECB, CBC, CTR</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Mode 4: Key derivation then decryption</td>
<td>ECB, CBC</td>
<td>145</td>
</tr>
</tbody>
</table>

Table 123. Processing latency for GCM and CCM (in clock cycles)

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Algorithm</th>
<th>Init Phase</th>
<th>Header phase(1)</th>
<th>Payload phase(1)</th>
<th>Tag phase(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</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>256-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>
</tbody>
</table>

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

22.7 AES registers

22.7.1 AES control register (AES_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>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.

Bits 23:20 **NPBLB[3:0]**: Number of padding bytes in last block

The bitfield sets the number of padding bytes in last block of payload:

- 0000: All bytes are valid (no padding)
- 0001: Padding for one least-significant byte of last block
- ...
- 1111: Padding for 15 least-significant bytes of last block

Bit 19 Reserved, must be kept at reset value.
Bit 18 **KEYSIZE**: Key size selection
   This bitfield defines the length of the key used in the AES cryptographic core, in bits:
   0: 128
   1: 256
   Attempts to write the bit are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMFH[1:0]**: GCM or CCM phase selection
   This bitfield selects the phase of GCM, GMAC or CCM algorithm:
   00: Init phase
   01: Header phase
   10: Payload phase
   11: Final phase
   The bitfield has no effect if other than GCM, GMAC or CCM algorithms are selected (through the ALGOMODE bitfield).

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).
   Use 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).
   Use 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 16, 6:5 **CHMOD[2:0]**: Chaining mode selection
This bitfield 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)
- others: Reserved
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

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
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access. 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 22.4.13: AES data registers and data swapping.
Attempts to write the bitfield are ignored when the EN bit of the AES_CR register is set before the write access and it is not cleared by that write access.

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 upon the completion of the key preparation (Mode 2) and upon the completion of GCM/GMAC/CCM initial phase.
22.7.2 AES status register (AES_SR)

Address offset: 0x04
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>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 31:4 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
When the flag indicates “idle”, the current GCM encryption processing may be suspended to process a higher-priority message. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored for the suspend process.

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. Unexpected write is ignored.

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. Unexpected read returns zero.

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.
22.7.3 AES data input register (AES_DINR)

Address offset: 0x08
Reset value: 0x0000 0000
Only 32-bit access type is supported.

Bits 31:0 DIN[31:0]: Input data word
A four-fold sequential write to this bitfield during the input phase results in writing a complete 128-bit block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit 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 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 22.4.13: AES data registers and data swapping on page 620.

22.7.4 AES data output register (AES_DOUTR)

Address offset: 0x0C
Reset value: 0x0000 0000
Only 32-bit read access type is supported.
Bits 31:0 **DOUT[31:0]:** Output data word

This read-only bitfield fetches a 32-bit output buffer. A four-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. Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [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 22.4.13: AES data registers and data swapping on page 620.

### 22.7.5 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>
</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 **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 KEYSIZE value is correct and when the AES peripheral is disabled (EN bit of the AES_CR register cleared). Note that, if, the key is directly loaded to AES_KEYRx registers (hence writes to key register is ignored and KEIF is set). Refer to Section 22.4.14: AES key registers on page 622 for more details.
### 22.7.6 AES key register 1 (AES_KEYR1)

Address offset: 0x14  
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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **KEY[63:48]**: Cryptographic key, bits [63:32]  
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

### 22.7.7 AES key register 2 (AES_KEYR2)

Address offset: 0x18  
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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>
<td></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.

### 22.7.8 AES key register 3 (AES_KEYR3)

Address offset: 0x1C  
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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]  
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.
### 22.7.9 AES initialization vector register 0 (AES_IVR0)

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>

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]  
Refer to Section 22.4.15: AES initialization vector registers on page 622 for description of the **IVI[127:0]** bitfield.  
The initialization vector is only used in chaining modes other than ECB.  
The AES_IVRx registers may be written only when the AES peripheral is disabled.

### 22.7.10 AES initialization vector register 1 (AES_IVR1)

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>
<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 **IVI[63:32]**: Initialization vector input, bits [63:32]  
Refer to the AES_IVR0 register for description of the **IVI[128:0]** bitfield.

### 22.7.11 AES initialization vector register 2 (AES_IVR2)

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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 **IVI[95:64]**: Initialization vector input, bits [95:64]  
Refer to the AES_IVR0 register for description of the **IVI[192:0]** bitfield.
### 22.7.12 AES initialization vector register 3 (AES_IVR3)

**Address offset:** 0x2C  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 IVI[95:64]: Initialization vector input, bits [95:64]</th>
<th>Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.</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>
</tr>
<tr>
<td>Bits 31:0 IVI[127:112]</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>Bits 31:0 IVI[111:96]</td>
<td></td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

### 22.7.13 AES key register 4 (AES_KEYR4)

**Address offset:** 0x30  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 IVI[127:96]: Initialization vector input, bits [127:96]</th>
<th>Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.</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>
</tr>
<tr>
<td>Bits 31:0 KEY[159:144]: Cryptographic key, bits [159:128]</td>
<td>Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
</tr>
<tr>
<td>Bits 31:0 KEY[143:128]: Cryptographic key, bits [143:128]</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

### 22.7.14 AES key register 5 (AES_KEYR5)

**Address offset:** 0x34  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 KEY[191:160]: Cryptographic key, bits [191:160]</th>
<th>Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.</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>
</tr>
<tr>
<td>Bits 31:0 KEY[175:160]: Cryptographic key, bits [175:160]</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>
22.7.15  AES key register 6 (AES_KEYR6)

Address offset: 0x38
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>KEY[223:208]</td>
</tr>
<tr>
<td>0x20</td>
<td>KEY[207:192]</td>
</tr>
</tbody>
</table>

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.

22.7.16  AES key register 7 (AES_KEYR7)

Address offset: 0x3C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>KEY[255:240]</td>
</tr>
<tr>
<td>0x20</td>
<td>KEY[239:224]</td>
</tr>
</tbody>
</table>

Bits 31:0 KEY[255:224]: Cryptographic key, bits [255:224]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield.

Note: The key registers from 4 to 7 are used only when the key length of 256 bits is selected. They have no effect when the key length of 128 bits is selected (only key registers 0 to 3 are used in that case).

22.7.17  AES suspend registers (AES_SUSPxR)

Address offset: 0x040 + 0x4 * x, (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.

Bits 31:0 **SUSP[31:0]**: AES suspend

Upon suspend operation, this bitfield of the corresponding AES_SUSPxR register takes the value of one of internal AES registers.

### 22.7.18 AES register map

#### Table 124. AES register map and reset values

<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>0x000</td>
<td>AES_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>
</tr>
<tr>
<td></td>
<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>
</tr>
<tr>
<td>0x004</td>
<td>AES_SR</td>
<td></td>
<td></td>
<td></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>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>
</tr>
<tr>
<td>0x008</td>
<td>AES_DINR</td>
<td>DIN[31:0]</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></td>
<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>
</tr>
<tr>
<td>0x00C</td>
<td>AES_DOOUTR</td>
<td>DOUT[31:0]</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></td>
<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>
</tr>
<tr>
<td>0x010</td>
<td>AES_KEYR0</td>
<td>KEY[31:0]</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></td>
<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>
</tr>
<tr>
<td>0x014</td>
<td>AES_KEYR1</td>
<td>KEY[32:0]</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></td>
<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>
</tr>
<tr>
<td>0x018</td>
<td>AES_KEYR2</td>
<td>KEY[95:64]</td>
<td></td>
<td></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>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>
</tr>
<tr>
<td>0x01C</td>
<td>AES_KEYR3</td>
<td>KEY[127:96]</td>
<td></td>
<td></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>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>
</tr>
<tr>
<td>0x020</td>
<td>AES_IVR0</td>
<td>IV[31:0]</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></td>
<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>
</tr>
<tr>
<td>0x024</td>
<td>AES_IVR1</td>
<td>IV[63:32]</td>
<td></td>
<td></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>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>
</tr>
<tr>
<td>0x028</td>
<td>AES_IVR2</td>
<td>IV[95:64]</td>
<td></td>
<td></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>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>
</tr>
</tbody>
</table>

---

636/1530 RM0434 Rev 13
Refer to Section 2.2 on page 66 for the register boundary addresses.
23 Public key accelerator (PKA)

23.1 Introduction
PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

All needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

23.2 PKA main features
- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA Chinese Remainder Theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check
  - ECDSA signature generation and verification
- Capability to handle operands up to 3136 bits for RSA/DH and 640 bits for ECC.
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication.
- Built-in Montgomery domain inward and outward transformations.
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise, for writes, an AHB bus error is generated, and write accesses are ignored).

23.3 PKA functional description

23.3.1 PKA block diagram

*Figure 143* shows the block diagram of the public key accelerator PKA.
23.3.2 PKA internal signals

Table 125 lists internal signals available at the IP level, not necessarily available on product bonding pads.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pka_hclk</td>
<td>Digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>pka_it</td>
<td>Digital output</td>
<td>Public key accelerator IP global interrupt request</td>
</tr>
</tbody>
</table>

23.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. The RAM receives this clock directly, the core is clocked at half the frequency.

23.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 3136 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation \( y^2 = x^3 + ax + b \) (mod p). More information is found in Section 23.5.1: Supported elliptic curves.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 3576 bytes (894 words of 32 bits) called PKA RAM is used for providing initial data to the PKA, and for holding the results after computation is completed. Access is done through the PKA AHB interface.
PKA operating modes

The list of operations the PKA can perform is detailed in Table 126 and Table 127, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register.

<table>
<thead>
<tr>
<th>PKA_CR.MODE[5:0]</th>
<th>Performed operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>Montgomery parameter computation R2 mod n</td>
<td>Section 23.4.2</td>
</tr>
<tr>
<td>0x0E</td>
<td>Modular addition (A+B) mod n</td>
<td>Section 23.4.3</td>
</tr>
<tr>
<td>0x0F</td>
<td>Modular subtraction (A-B) mod n</td>
<td>Section 23.4.4</td>
</tr>
<tr>
<td>0x10</td>
<td>Montgomery multiplication (AxB) mod n</td>
<td>Section 23.4.5</td>
</tr>
<tr>
<td>0x00</td>
<td>Modular exponentiation A^e mod n</td>
<td>Section 23.4.6</td>
</tr>
<tr>
<td>0x02</td>
<td>Modular exponentiation A^e mod n (fast mode)</td>
<td>Section 23.4.7</td>
</tr>
<tr>
<td>0x08</td>
<td>Modular inversion A-1 mod n</td>
<td>Section 23.4.8</td>
</tr>
<tr>
<td>0x0D</td>
<td>Modular reduction A mod n</td>
<td>Section 23.4.9</td>
</tr>
<tr>
<td>0x09</td>
<td>Arithmetic addition A+B</td>
<td>Section 23.4.10</td>
</tr>
<tr>
<td>0x0A</td>
<td>Arithmetic subtraction A-B</td>
<td>Section 23.4.11</td>
</tr>
<tr>
<td>0x0B</td>
<td>Arithmetic multiplication AxB</td>
<td>Section 23.4.12</td>
</tr>
<tr>
<td>0x0C</td>
<td>Arithmetic comparison (A=B, A&gt;B, A&lt;B)</td>
<td>Section 23.4.13</td>
</tr>
<tr>
<td>0x07</td>
<td>RSA CRT exponentiation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PKA_CR.MODE[5:0]</th>
<th>Performed operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x28</td>
<td>Point on elliptic curve Fp check</td>
<td>Section 23.4.14</td>
</tr>
<tr>
<td>0x20</td>
<td>ECC scalar multiplication kP</td>
<td>Section 23.4.15</td>
</tr>
<tr>
<td>0x22</td>
<td>ECC scalar multiplication kP (fast mode)</td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>ECDSA sign</td>
<td>Section 23.4.16</td>
</tr>
<tr>
<td>0x26</td>
<td>ECDSA verification</td>
<td>Section 23.4.17</td>
</tr>
</tbody>
</table>

Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is
supplied before starting the operation. Performance improvement is detailed in Section 23.5.2: Computation times.

The operations using fast mode are modular exponentiation and scalar multiplication.

23.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:
- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:
- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

RSA key pair

For following RSA operations a public key and a private key information are defined as below:
- Alice transmits her public key \((n, e)\) to Bob. Numbers \(n\) and \(e\) are very large positive integers.
- Alice keeps secret her private key \(d\), also a very large positive integer. Alternatively this private key can also be represented by a quintuple \((p, q, dp, dq, qInv)\).

For more information on above representations refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message \(M\) using Alice’s public key \((n, e)\) must go through the following steps:
1. Compute the encoded message \(EM = ENCODE(M)\), where ENCODE is an encoding method.
2. Turn \(EM\) into an integer \(m\), with \(0 \leq m < n\) and \((m, n)\) being co-primes.
3. Compute ciphertext \(c = m^e \mod n\).
4. Convert the integer \(c\) into a string ciphertext \(C\).
Alice, to decrypt ciphertext c using her private key, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.

2. Recover plaintext \( m = c^d \mod n = (m^e)^d \mod n \). If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext \( m \) is obtained by performing the operations:
   a) \( m_1 = c^{dp} \mod p \)
   b) \( m_2 = c^{dq} \mod q \)
   c) \( h = q^{-1}(m_1 - m_2) \mod p \)
   d) \( m = m_2 + h \cdot q \)

3. Convert the integer message representative \( m \) to an encoded message EM.

4. Recover message \( M = \text{DECODE}(EM) \), where DECODE is a decoding method.

Above operations can be accelerated by PKA using Modular exponentiation \( A^e \mod n \) if the private key is \( d \), or RSA CRT exponentiation if the private key is the quintuple (p, q, dp, dq, qInv).

**Note:** The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

**Elliptic curve selection**

For following ECC operations curve parameters are defined as below:
- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob).
- Supported curves parameters are summarized in Section 23.5.1: Supported elliptic curves.
- \( G \) is the chosen elliptic curve base point (also known as generator), with a large prime order \( n \) (i.e. \( n \cdot G = \text{identity element O} \)).

**ECDSA message signature generation**

ECDSA (Elliptic Curve Digital Signature Algorithm) signature generation function principle is the following: Alice, to sign a message \( m \) using her private key integer \( d_A \), follows the steps below.

1. Calculate \( e = \text{HASH}(m) \), where HASH is a cryptographic hash function.
2. Let \( z \) be the \( L_n \) leftmost bits of \( e \), where \( L_n \) is the bit length of the group order \( n \).
3. Select a cryptographically secure random integer \( k \) where \( 0 < k < n \).
4. Calculate the curve point \( (x_1, y_1) = k \cdot G \).
5. Calculate \( r = x_1 \mod n \). If \( r = 0 \) go back to step 3.
6. Calculate \( s = k^{-1}(z + rd_A) \mod n \). If \( s = 0 \) go back to step 3.
7. The signature is the pair \( (r, s) \).

Steps 4 to 7 are accelerated by PKA using:

**ECDSA sign** or

All of the operations below:
- **ECC Fp scalar multiplication** \( k \cdot P \)
- **Modular reduction** \( A \mod n \)
- **Modular inversion** \( A^{-1} \mod n \)
- **Modular addition** and **Modular and Montgomery multiplication**
ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point $Q_A$.

Bob can verify that $Q_A$ is a valid curve point going through the following steps:
1. check that $Q_A$ is not equal to the identity element $O$
2. check that $Q_A$ is on the agreed curve
3. check that $n \times Q_A = O$.

Then Bob follows the procedure detailed below:
1. verify that $r$ and $s$ are integer in $[1, n-1]$
2. calculate $e = \text{HASH}(m)$, where $\text{HASH}$ is the agreed cryptographic hash function
3. let $z$ be the $L_n$ leftmost bits of $e$
4. calculate $w = s^{-1} \mod n$
5. calculate $u_1 = zw \mod n$ and $u_2 = rw \mod n$
6. calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. the signature is valid if $r = x_1 \pmod{n}$, it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using ECDSA verification.

23.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in Section 23.4.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:
1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy ($BUSY = 1$) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.
Using precomputed Montgomery parameters (PKA fast mode)

As explained in Section 23.3.4, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, Section 23.4.5). This is know as “fast mode”.

To manage Fast Mode usage the recommended procedure is described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in Section 23.5.1.

2. Program in PKA_CR register the PKA in Montgomery parameter computation mode (MODE=“0x1”) then assert the START bit.

3. Wait until the PROCENDF bit in the PKA_SR register is set to “1”, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in Section 23.4.

23.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the Address Error flag (ADDRERRF) is set in the PKA_SR register.

- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM Error Flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return zero, while writes are ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see Section 23.6 for details).

ADDRERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

23.4 PKA operating modes

23.4.1 Introduction

The various operations supported by PKA are described in the following subsections, clarifying the associated format of the input data and of the results, both stored in the PKA RAM.

The following information applies to all PKA operations.

PKA core processes 32-bit words

Supported operand “Size” are:

- ROS (RSA operand size): data size is \(\frac{rsa\_size}{32}+1\) words, with \(rsa\_size\) equal to the chosen modulus length. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 33 words, or 1056 bits.

- EOS (ECC operand size): data size is \(\frac{ecc\_size}{32}+1\) words, with \(ecc\_size\) equal to the chosen prime modulus length. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 7 words, or 224 bits.
Note: Fractional results for above formulas are rounded up to the nearest integer since PKA core processes 32-bit words.

Note: The maximum ROS is 99 words (3136-bit max exponent size), while the maximum EOS is 21 words (640-bit max operand size).

The column indicated with Storage in the following tables indicates either the Register PKA_CR, or the address offset within the PKA, located in the PKA RAM area (starting from 0x400). To recover the physical address of an operand the application must add to the indicated offset the base address of the PKA.

About writing parameters ("IN" direction)
- When elements are written as input in the PKA memory, an additional word with all bits equal to zero must be added. As an example, when ECC P256 is used and when loading an input (represented on 256 bits or 8 words), an additional word is expected by the PKA and it has to be filled with zeros.
- About endianess, for example to prepare the operation ECC Fp scalar multiplication, when application writes $x_p$ coordinate for an ECC P256 curve (EOS= 9 words) the least significant bit is to be placed in bit 0 at address offset 0x55C; the most significant bit is to be placed in bit 31 of address offset 0x578. Then, as mentioned above, word 0x00000000 should also be written at address offset 0x57C.
- Unless indicated otherwise all operands in the tables are integers.

About PKA outputs ("OUT" direction)
- Unless specified in the tables PKA does not provide an error output when a wrong parameter is written by the application.

Caution: Validity of all input parameters to the PKA must be checked before issuing any PKA operation. Indeed, the PKA assumes that all input parameters are valid and consistent with each other.

23.4.2 Montgomery parameter computation
This function is used to compute the Montgomery parameter ($R^2 \mod n$) used by PKA to convert operands into the Montgomery residue system representation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in Table 128.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MODE</td>
<td>0x01</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Modulus length (In bits, 0 ≤ value &lt; 3136bits)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
<td></td>
</tr>
<tr>
<td>Modulus value n (Odd integer only, n &lt; 2^{3136})</td>
<td>RAM@0xD5C</td>
<td>ROS</td>
<td></td>
</tr>
<tr>
<td>OUT Result: $R^2 \mod n$</td>
<td>-</td>
<td>RAM@0x594</td>
<td></td>
</tr>
</tbody>
</table>
23.4.3 Modular addition

Modular addition operation consists in the computation of \( A + B \mod n \). Operation instructions are summarized in Table 129.

Table 129. Modular addition

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0E</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>( 0 \leq A &lt; n )</td>
<td>RAM@0x8B4</td>
<td></td>
</tr>
<tr>
<td>Operand B</td>
<td>( 0 \leq B &lt; n )</td>
<td>RAM@0xA44</td>
<td></td>
</tr>
<tr>
<td>Modulus value n</td>
<td>( n &lt; 2^{3136} )</td>
<td>RAM@0xD5C</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: ( A+B \mod n )</td>
<td>(0 ≤ result &lt; n)</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

23.4.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If \( A \geq B \) result equals \( A - B \mod n \)
- If \( A < B \) result equals \( A + n - B \mod n \)

Operation instructions are summarized in Table 130.

Table 130. Modular subtraction

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0F</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>( 0 \leq A &lt; n )</td>
<td>RAM@0x8B4</td>
<td></td>
</tr>
<tr>
<td>Operand B</td>
<td>( 0 \leq B &lt; n )</td>
<td>RAM@0xA44</td>
<td></td>
</tr>
<tr>
<td>Modulus value n</td>
<td>( n &lt; 2^{3136} )</td>
<td>RAM@0xD5C</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: ( A-B \mod n )</td>
<td>(0 ≤ result &lt; n)</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

23.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication \( A \times B \mod n \)

The method to perform above operations are described below. Note that “\( x \)” function is this operation, and A, B, C operands are in the natural domain.
1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Let’s assume A is an integer in the natural domain
      Compute \( r2modn \) using Montgomery parameter computation
      Result \( AR = A \times r2modn \mod n \) is A in the Montgomery domain
   b) Let’s assume \( BR \) is an integer in the Montgomery domain
      Result \( B = BR \times 1 \mod n \) is B in the natural domain
      Similarly, above value \( AR \) computed in a) can be converted into the natural domain by computing \( A = AR \times 1 \mod n \)

2. Simple modular multiplication \( A \times B \mod n \)
   a) Compute \( r2modn \) using Montgomery parameter computation
   b) Compute \( AR = A \times r2modn \mod n \). Output is in the Montgomery domain
   c) Compute \( AB = AR \times B \mod n \). Output is in natural domain

3. Multiple modular multiplication \( A \times B \times C \mod n \)
   a) Compute \( r2modn \) using Montgomery parameter computation
   b) Compute \( AR = A \times r2modn \mod n \). Output is in the Montgomery domain
   c) Compute \( BR = B \times r2modn \mod n \). Output is in the Montgomery domain
   d) Compute \( ABR = AR \times B \mod n \). Output is in the Montgomery domain
   e) Compute \( CR = C \times r2modn \mod n \). Output is in the Montgomery domain
   f) Compute \( ABCR = ABR \times CR \mod n \). Output is in the Montgomery domain
   g) (optional) Repeat the two steps above if more operands need to be multiplied
   h) Compute \( ABC = ABCR \times 1 \mod n \) to retrieve the result in natural domain

Operation instructions for Montgomery multiplication are summarized in Table 131.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE</td>
<td>0x10</td>
<td>PKA_CR</td>
</tr>
<tr>
<td></td>
<td>Operand length</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
</tr>
<tr>
<td></td>
<td>Operand A</td>
<td>(0 ≤ A &lt; n)</td>
<td>RAM@0x8B4</td>
</tr>
<tr>
<td></td>
<td>Operand B</td>
<td>(0 ≤ B &lt; n)</td>
<td>RAM@0xA44</td>
</tr>
<tr>
<td></td>
<td>Modulus value n</td>
<td>(Odd integer only, n &lt; 2^{3136})</td>
<td>RAM@0xDSC</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: AxB mod n(1)</td>
<td>-</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples 2 and 3).

23.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of \( A^n \mod n \).

Operation instructions for modular exponentiation are summarized in Table 132 (normal mode) and in Table 133 (fast mode). Fast mode usage is explained in Section 23.3.6.
23.4.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse \( A^{-1} \mod n \). If the modulus \( n \) is prime, for all values of \( A (1 \leq A < n) \) modular inversion output is valid. If the modulus \( n \) is not prime, \( A \) has an inverse only if the largest common divisor between \( A \) and \( n \) is 1.

If the operand \( A \) is a divisor of the modulus \( n \), the result is a multiple of a factor of \( n \).

Operation instructions for modular inversion are summarized in Table 134.

### Table 134. Modular inversion

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE</td>
<td>0x08</td>
<td>PKA_CR</td>
</tr>
<tr>
<td>IN</td>
<td>Operand length</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
</tr>
<tr>
<td>IN</td>
<td>Operand A (base of exponentiation)</td>
<td>( 0 \leq A &lt; n )</td>
<td>RAM@0xA44</td>
</tr>
<tr>
<td>IN</td>
<td>Exponent e</td>
<td>( 0 \leq e &lt; n )</td>
<td>RAM@0xBD0</td>
</tr>
<tr>
<td>IN</td>
<td>Modulus value n</td>
<td>(Odd integer only, ( n &lt; 2^{3136} ))</td>
<td>RAM@0xD5C</td>
</tr>
<tr>
<td>IN/OOUT</td>
<td>Montgomery param R2 mod n</td>
<td>(mandatory)</td>
<td>RAM@0x594</td>
</tr>
<tr>
<td>OUT</td>
<td>Result: ( A^{-1} \mod n )</td>
<td>( 0 &lt; ) result &lt; ( n )</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

---

648/1530  RM0434 Rev 13
### 23.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of $A$ divided by $n$. Operation instructions are summarized in Table 135.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0D</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length</td>
<td>(In bits, not null)</td>
<td>RAM@0x400</td>
<td>32 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(In bits, 8 &lt; value &lt; 3136)</td>
<td>RAM@0x404</td>
<td></td>
</tr>
<tr>
<td>Operand A</td>
<td>($0 \leq A &lt; 2n &lt; 2^{3136}$)</td>
<td>RAM@0x8B4</td>
<td></td>
</tr>
<tr>
<td>Modulus value $n$</td>
<td>(Odd integer only, $n &lt; 2^{3136}$)</td>
<td>RAM@0xA44</td>
<td>ROS</td>
</tr>
<tr>
<td>OUT</td>
<td>Result A mod n</td>
<td>(0 &lt; result &lt; n)</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

### 23.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of $A + B$. Operation instructions are summarized in Table 136.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x09</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>($0 \leq A &lt; 2^M$)</td>
<td>RAM@0x8B4</td>
<td></td>
</tr>
<tr>
<td>Operand B</td>
<td>($0 \leq B &lt; 2^M$)</td>
<td>RAM@0xA44</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result $A+B$</td>
<td>(0 ≤ result &lt; $2^{M+1}$)</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

### 23.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If $A \geq B$ result equals $A - B$
- If $A < B$ and $M/32$ residue is $>0$ result equals $A + 2^\lceil(M/32)\rceil 32 + 1 - B$
- If $A < B$ and $M/32$ residue is $0$ result equals $A + 2^\lceil(M/32)\rceil 32 - B$

Operation instructions are summarized in Table 137.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0A</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>($0 \leq A &lt; 2^M$)</td>
<td>RAM@0x8B4</td>
<td></td>
</tr>
<tr>
<td>Operand B</td>
<td>($0 \leq B &lt; 2^M$)</td>
<td>RAM@0xA44</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result $A-B$</td>
<td>($0 \leq result &lt; 2^M$)</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>
23.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in Table 138.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0B</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>(0 ≤ A &lt; 2^M)</td>
<td>RAM@0xB4</td>
<td>ROS</td>
</tr>
<tr>
<td>Operand B</td>
<td>(0 ≤ B &lt; 2^M)</td>
<td>RAM@0xA44</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: AxB</td>
<td>(0 ≤ result &lt; 2^M)</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

23.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:
- If A=B then result=0x0
- If A>B then result=0x1
- If A<B then result=0x2

Operation instructions for arithmetic comparison are summarized in Table 139.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0C</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>(0 ≤ A &lt; 2^M)</td>
<td>RAM@0xB4</td>
<td>ROS</td>
</tr>
<tr>
<td>Operand B</td>
<td>(0 ≤ B &lt; 2^M)</td>
<td>RAM@0xA44</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result A=B or A&gt;B or A&lt;B</td>
<td>0x0, 0x1 or 0x02</td>
<td>RAM@0xBD0</td>
</tr>
</tbody>
</table>

23.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries like OpenSSL RSA use the following optimization for decryption and signing based on the Chinese remainder theorem (CRT):
- p and q are precomputed primes, stored as part of the private key
- \(d_p = d \mod (p - 1)\)
- \(d_q = d \mod (q - 1)\) and
- \(q_{\text{inv}} = q^{-1} \mod p\)
These values allow the recipient to compute the exponentiation \( m = A^d \mod pq \) more efficiently as follows:

- \( m_1 = A^{dP} \mod p \)
- \( m_2 = A^{dQ} \mod p \)
- \( h = q_{\text{inv}} \mod p \), with \( m_1 > m_2 \)
- \( m = m_2 + hQ \)

Operation instructions for computing CRT exponentiation \( A^d \mod pq \) are summarized in Table 140.

#### Table 140. CRT exponentiation

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MODE</td>
<td>0x07</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>IN Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>IN Operand ( d_P )</td>
<td>( 0 \leq d_P &lt; 2^{M/2} )</td>
<td>RAM@0x65C</td>
<td></td>
</tr>
<tr>
<td>IN Operand ( d_Q )</td>
<td>( 0 \leq d_Q &lt; 2^{M/2} )</td>
<td>RAM@0xBD0</td>
<td></td>
</tr>
<tr>
<td>IN ( q_{\text{inv}} )</td>
<td>( 0 \leq q_{\text{inv}} &lt; 2^{M/2} )</td>
<td>RAM@0x7EC</td>
<td>ROS/2</td>
</tr>
<tr>
<td>Prime ( p^{(1)} )</td>
<td>( 0 \leq p &lt; 2^{M/2} )</td>
<td>RAM@0x97C</td>
<td></td>
</tr>
<tr>
<td>Prime ( q^{(1)} )</td>
<td>( 0 \leq q &lt; 2^{M/2} )</td>
<td>RAM@0xD5C</td>
<td></td>
</tr>
<tr>
<td>IN Operand ( A )</td>
<td>( 0 \leq A &lt; 2^{M/2} )</td>
<td>RAM@0xEEC</td>
<td></td>
</tr>
<tr>
<td>OUT Result: ( A^d \mod pq )</td>
<td>( 0 \leq \text{result} &lt; pq )</td>
<td>RAM@0x724</td>
<td>ROS</td>
</tr>
</tbody>
</table>

1. Must be different from 2.

### 23.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point \( P (x, y) \) satisfies or not the curves over prime fields equation \( y^2 = (x^3 + ax + b) \mod p \), where \( a \) and \( b \) are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in Table 141.
23.4.15 ECC Fp scalar multiplication

This operation consists in the computation of a k x P (xP, yP), where P is a point on a curve over prime fields and "x" is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in Table 142 (normal mode) and Table 143 (fast mode). Fast mode usage is explained in Section 23.3.6.

Table 141. Point on elliptic curve Fp check

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td>0x28 PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(In bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>0x0: positive</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient [a]</td>
<td>(Absolute value, [a] &lt; p)</td>
<td>RAM@0x40C</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient b</td>
<td>([b] &lt; p)</td>
<td>RAM@0x7FC</td>
<td>EOS</td>
</tr>
<tr>
<td>Curve modulus value p</td>
<td>(Odd integer prime, 0 &lt; p &lt; 2640)</td>
<td>RAM@0x460</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate x</td>
<td>(x &lt; p)</td>
<td>RAM@0x55C</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate y</td>
<td>(y &lt; p)</td>
<td>RAM@0x5B0</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: P on curve</td>
<td>0x0: point on curve</td>
<td>RAM@0x400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not 0x0: point not on curve</td>
<td>RAM@0x5B0</td>
</tr>
</tbody>
</table>

Table 142. ECC Fp scalar multiplication

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td>0x20 PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Scalar multiplier k length</td>
<td>(In bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x400</td>
<td>32 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(In bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x404</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>0x0: positive</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient [a]</td>
<td>(Absolute value, [a] &lt; p)</td>
<td>RAM@0x40C</td>
<td></td>
</tr>
<tr>
<td>Curve modulus value p</td>
<td>(Odd integer prime, 0 &lt; p &lt; 2540)</td>
<td>RAM@0x460</td>
<td>EOS</td>
</tr>
<tr>
<td>Scalar multiplier k</td>
<td>(0 ≤ k &lt; 2540)</td>
<td>RAM@0x508</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate x</td>
<td>(x &lt; p)</td>
<td>RAM@0x55C</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate y</td>
<td>(y &lt; p)</td>
<td>RAM@0x5B0</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: k x P coordinate x</td>
<td>(result &lt; p)</td>
<td>RAM@0x5B0</td>
</tr>
<tr>
<td></td>
<td>Result: k x P coordinate y</td>
<td>(result &lt; p)</td>
<td>RAM@0x5B0</td>
</tr>
</tbody>
</table>
When performing this operation following special cases should be noted:

For k = 0, this function returns a point at infinity, that is (0, 0) if curve parameter b is nonzero and (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation should be carried out.

For k < 0 (i.e. a negative scalar multiplication is required) multiplier absolute value k = |−k| should be provided to the PKA. After the computation completion, the formula −P = (x, −y) can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

### 23.4.16 ECDSA sign

ECDSA signing operation (outlined in Section 23.3.5) is summarized in Table 144 (input parameters) and in Table 145 (output parameters).

The application should check if the output error is equal to zero, if it is different from zero a new k should be generated and the ECDSA sign operation should be repeated.
Note: If error output is different from zero the content of the PKA memory should be cleared to avoid leaking information about the private key.

Extended ECDSA support

PKA also supports Extended ECDSA signature, for which the inputs and the outputs have the same ECDSA signature (Table 144 and Table 145, respectively), with the addition of the coordinates of the point kG. This extra output is defined in Table 146.
23.4.17 ECDSA verification

ECDSA verification operation (outlined in Section 23.3.5) is summarized in Table 147 (input parameters) and Table 148 (output parameters).

The application should check if the output error is equal to zero, if it is different from zero, the signature is not verified.

Table 147. ECDSA verification (inputs)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x26</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve prime order n length</td>
<td>(In bits, not null)</td>
<td>RAM@0x404</td>
<td>32 bits</td>
</tr>
<tr>
<td>Curve modulus p length</td>
<td>(In bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x4B4</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>0x0: positive</td>
<td>RAM@0x45C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1: negative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curve coefficient</td>
<td>[a]</td>
<td>(Absolute value,</td>
<td>RAM@0x460</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[a] &lt; p)</td>
<td></td>
</tr>
<tr>
<td>Curve modulus p value</td>
<td>(Odd integer prime, 0 &lt; p &lt; 2^{640})</td>
<td>RAM@0x4B8</td>
<td></td>
</tr>
<tr>
<td>Curve base point G x</td>
<td>(x &lt; p)</td>
<td>RAM@0x5E8</td>
<td></td>
</tr>
<tr>
<td>Curve base point G y</td>
<td>(y &lt; p)</td>
<td>RAM@0x63C</td>
<td></td>
</tr>
<tr>
<td>Public-key curve point Q</td>
<td>(xQ &lt; p)</td>
<td>RAM@0xF40</td>
<td></td>
</tr>
<tr>
<td>coordinate xQ</td>
<td></td>
<td></td>
<td>EOS</td>
</tr>
<tr>
<td>Public-key curve point Q</td>
<td>(yQ &lt; p)</td>
<td>RAM@0xF94</td>
<td></td>
</tr>
<tr>
<td>coordinate yQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signature part r</td>
<td>(0 &lt; r &lt; n)</td>
<td>RAM@0x108</td>
<td></td>
</tr>
<tr>
<td>Signature part s</td>
<td>(0 &lt; s &lt; n)</td>
<td>RAM@0x4A4</td>
<td></td>
</tr>
<tr>
<td>Hash of message z</td>
<td>(z &lt; 2^M)</td>
<td>RAM@0xF8E</td>
<td></td>
</tr>
<tr>
<td>Curve prime order n</td>
<td>(integer prime)</td>
<td>RAM@0xD5C</td>
<td></td>
</tr>
</tbody>
</table>

Table 148. ECDSA verification (outputs)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (Note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result: ECDSA verify</td>
<td>0x0: valid signature</td>
<td>RAM@0x5B0</td>
<td>32 bits</td>
</tr>
<tr>
<td></td>
<td>Not 0x0: invalid signature</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
23.5 Example of configurations and processing times

23.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation \( y^2 = x^3 + ax + b \pmod{p} \).

*Note:* Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer \( p \), used as the modulus for all point arithmetic in the finite field \( \text{GF}(p) \)
- the (usually prime) integer \( n \), the order of the group generated by \( G \), defined below
- the base point of the curve \( G \), defined by its coordinates \( (G_x, G_y) \)
- the integers \( a \) and \( b \), coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define \( a \) as negative, PKA supports two representations:

1. **a defined as \( p-|a| \)** in the finite field \( \text{GF}(p) \), for example \( p-3 \):
   
   Curve coefficient \( p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF \)
   Curve coefficient \( a \) sign= 0x0 (positive)
   Curve coefficient \( a = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF \)

2. **a defined as negative**, for example -3:
   
   Curve coefficient \( p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF \)
   Curve coefficient \( a \) sign= 0x1 (negative)
   Curve coefficient \( a = 0x00000000 00000000 00000000 00000000 00000000 00000001 00000000 00000003 \)

*Table 149* summarizes the family of curves supported by PKA for ECC operations.

<table>
<thead>
<tr>
<th>Curve name</th>
<th>Standard</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-192</td>
<td>NIST</td>
<td><em>Digital Signature Standard (DSS)</em>, NIST FIPS 186-4</td>
</tr>
<tr>
<td>P-224</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-521</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 149. Family of supported curves for ECC operations (continued)

<table>
<thead>
<tr>
<th>Curve name</th>
<th>Standard</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>brainpoolP224r1, brainpoolP224t1</td>
<td></td>
<td>– Brainpool Elliptic Curves, IETF RFC 5639</td>
</tr>
<tr>
<td>brainpoolP256r1, brainpoolP256t1</td>
<td>IETF</td>
<td>– Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry, IETF RFC 6932</td>
</tr>
<tr>
<td>brainpoolP320r1, brainpoolP320t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP384r1, brainpoolP384t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP512r1, brainpoolP512t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp192k1, secp192r1</td>
<td>SEC</td>
<td>Standards for Efficient Cryptography SEC 2 curves</td>
</tr>
<tr>
<td>secp224k1, secp224r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp256k1, secp256r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp384r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp521r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recommended curve parameters for public key cryptographic algorithm SM2</td>
<td>OSCCA</td>
<td>– Public key cryptographic algorithm SM2 based on elliptic curves, Organization of State Commercial Administration of China OSCCA SM2, December 2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Digital signatures - Part 3 Discrete logarithm based mechanisms, ISO/IEC 14888-3, November 2018</td>
</tr>
</tbody>
</table>
### 23.5.2 Computation times

The following tables summarize the PKA computation times, expressed in clock cycles.

#### Table 150. Modular exponentiation computation times

<table>
<thead>
<tr>
<th>Exponent length (in bits)</th>
<th>Mode</th>
<th>Modulus length (in bits)</th>
<th>1024</th>
<th>2048</th>
<th>3072</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>304000</td>
<td>814000</td>
<td>1728000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>46000</td>
<td>164000</td>
<td>356000</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Normal</td>
<td>326000</td>
<td>896000</td>
<td>1910000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>68000</td>
<td>246000</td>
<td>534000</td>
<td></td>
</tr>
<tr>
<td>(2^{16} + 1)</td>
<td>Normal</td>
<td>416000</td>
<td>1222000</td>
<td>2616000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>158000</td>
<td>572000</td>
<td>1244000</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>Normal</td>
<td>11664000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>11280000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CRT(^{(1)})</td>
<td></td>
<td>3546000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td>Normal</td>
<td>-</td>
<td>83834000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>-</td>
<td>82046000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CRT(^{(1)})</td>
<td></td>
<td>-</td>
<td>23468000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>3072</td>
<td>Normal</td>
<td>-</td>
<td>-</td>
<td>274954000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>-</td>
<td>-</td>
<td>273522000</td>
<td></td>
</tr>
<tr>
<td>CRT(^{(1)})</td>
<td></td>
<td>-</td>
<td>-</td>
<td>73378000</td>
<td></td>
</tr>
</tbody>
</table>

1. CRT stands for chinese remainder theorem optimization (MODE bitfield = 0x07).

#### Table 151. ECC scalar multiplication computation times\(^{(1)}\)

<table>
<thead>
<tr>
<th>Mode</th>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>1634000</td>
<td>2500000</td>
<td>4924000</td>
<td>8508000</td>
<td>13642000</td>
<td>28890000</td>
<td>33160000</td>
<td></td>
</tr>
<tr>
<td>Fast</td>
<td>1630000</td>
<td>2494000</td>
<td>4916000</td>
<td>8494000</td>
<td>13614000</td>
<td>28842000</td>
<td>33158000</td>
<td></td>
</tr>
</tbody>
</table>

1. These times depend on the number of "1"s included in the scalar parameter.

#### Table 152. ECDSA signature average computation times\(^{(1)}\)\(^{(2)}\)

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
</tr>
</thead>
<tbody>
<tr>
<td>1760000</td>
<td>2664000</td>
<td>5249000</td>
<td>9016000</td>
<td>14596000</td>
<td>30618000</td>
<td>35540000</td>
<td></td>
</tr>
</tbody>
</table>

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.

2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.
There are three individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. access to unmapped address (ADDRERRF), see Section 23.3.7
2. PKA RAM access while PKA operation is in progress (RAMERRF), see Section 23.3.7
3. PKA end of operation (PROCENDF)

The three interrupt sources are connected to the same global interrupt request signal pka_it. The user can enable or disable above interrupt sources individually by changing the mask bits in the PKA control register (PKA_CR). Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 156 gives a summary of the available features.

<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>PKA</td>
<td>Access to unmapped address error</td>
<td>ADDRERRF</td>
<td>ADDRRIE</td>
<td>Set ADDRERRFC bit</td>
</tr>
<tr>
<td></td>
<td>PKA RAM access error</td>
<td>RAMERRF</td>
<td>RAMRIE</td>
<td>Set RAMERRFC bit</td>
</tr>
<tr>
<td></td>
<td>PKA end of operation</td>
<td>PROCENDF</td>
<td>PROCENIE</td>
<td>Set PROCENDFC bit</td>
</tr>
</tbody>
</table>

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.
23.7 PKA registers

23.7.1 PKA control register (PKA_CR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Bit 15:10</th>
<th>Bit 9:4</th>
<th>Bit 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRERRIE</td>
<td>RAMERRIE</td>
<td>PROCENDIE</td>
<td>START EN</td>
</tr>
<tr>
<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 ADDRERRIE: Address error interrupt enable
  0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
  1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 RAMERRIE: RAM error interrupt enable
  0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
  1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 PROCENDIE: End of operation interrupt enable
  0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
  1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 MODE[5:0]: PKA operation code
  000000: Montgomery parameter computation then modular exponentiation
  000001: Montgomery parameter computation only
  000010: Modular exponentiation only (Montgomery parameter must be loaded first)
  100000: Montgomery parameter computation then ECC scalar multiplication
  100010: ECC scalar multiplication only (Montgomery parameter must be loaded first)
  100100: ECDSA sign
  100110: ECDSA verification
  101000: Point on elliptic curve Fp check
  000111: RSA CRT exponentiation
  001000: Modular inversion
  001001: Arithmetic addition
  001010: Arithmetic subtraction
  001011: Arithmetic multiplication
  001100: Arithmetic comparison
  001101: Modular reduction
  001110: Modular addition
  001111: Modular subtraction
  010000: Montgomery multiplication
  Others: Reserved
23.7.2 PKA status register (PKA_SR)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:21</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 20</td>
<td>ADDRERRF: Address error flag</td>
</tr>
<tr>
<td>0: No Address error</td>
<td></td>
</tr>
<tr>
<td>1: Address access is out of range (unmapped address)</td>
<td></td>
</tr>
<tr>
<td>This bit is cleared using ADDRERRFC bit in PKA_CLRFR.</td>
<td></td>
</tr>
<tr>
<td>Bit 19</td>
<td>RAMERRF: PKA RAM error flag</td>
</tr>
<tr>
<td>0: No PKA RAM access error</td>
<td></td>
</tr>
<tr>
<td>1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).</td>
<td></td>
</tr>
<tr>
<td>This bit is cleared using RAMERRFC bit in PKA_CLRFR.</td>
<td></td>
</tr>
<tr>
<td>Bit 18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 17</td>
<td>PROCENDF: PKA End of Operation flag</td>
</tr>
<tr>
<td>0: Operation in progress</td>
<td></td>
</tr>
<tr>
<td>1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.</td>
<td></td>
</tr>
<tr>
<td>Bit 16</td>
<td>BUSY: PKA operation is in progress</td>
</tr>
<tr>
<td>This bit is set to 1 whenever START bit in the PKA_CR is set. It is automatically cleared when the computation is complete, meaning that PKA RAM can be safely accessed and a new operation can be started.</td>
<td></td>
</tr>
<tr>
<td>0: No operation is in progress (default)</td>
<td></td>
</tr>
<tr>
<td>1: An operation is in progress</td>
<td></td>
</tr>
<tr>
<td>If PKA is started with a wrong opcode the peripheral is busy for a couple of cycles, then it aborts automatically the operation and go back to ready (BUSY bit is set to 0).</td>
<td></td>
</tr>
<tr>
<td>Bit 15:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
23.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08
Reset value: 0x0000 0000

![Table showing PKA_CLRFR bits]

Bits 31:21 Reserved, must be kept at reset value.

- **Bit 20 ADDRERRFC**: Clear Address error flag
  - 0: No action
  - 1: Clear the ADDRERRF flag in PKA_SR

- **Bit 19 RAMERRFC**: Clear PKA RAM error flag
  - 0: No action
  - 1: Clear the RAMERRF flag in PKA_SR

- **Bit 18 Reserved, must be kept at reset value.**

- **Bit 17 PROCENDFC**: Clear PKA End of Operation flag
  - 0: No action
  - 1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

*Note: Reading PKA_CLRFR returns all 0s.*

23.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 3576 bytes (max word offset: 0x11F4).
### 23.7.5 PKA register map

#### Table 157. PKA register map and reset values

| Offset | Register name | Offset | 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  | PKA_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  | 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  |
| 0x004  | PKA_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  | 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  |
| 0x008  | PKA_CLRFR     | 0      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 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  |

Refer to Section 2.2 on page 66 for the register boundary addresses.
24 Advanced-control timer (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.

24.1 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 24.3.26: Timer synchronization.
24.2 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 8.2.11: Clock security system (CSS) on HSE
   - A PVD output
   - SRAM parity error signal
   - CPU1 Cortex®-M4 LOCKUP (Hardfault) output.
   - COMPx output, x = 1,2.
24.3 TIM1 functional description

24.3.1 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 145* and *Figure 146* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 145. Counter timing diagram with prescaler division change from 1 to 2

CK_PSC

CEN

Timer clock = CK_CNT

Counter register: F7 F8 F9 FA FB FC 00 01 02 03

Update event (UEV)

Prescaler control register: 0 1

Write a new value in TIMx_PSC

Prescaler buffer: 0 1

Prescaler counter: 0 0 1 0 1 0 1 0 1

Figure 146. Counter timing diagram with prescaler division change from 1 to 4

CK_PSC

CEN

Timer clock = CK_CNT

Counter register: F7 F8 F9 FA FB FC 00 01

Update event (UEV)

Prescaler control register: 0 3

Write a new value in TIMx_PSC

Prescaler buffer: 0 3

Prescaler counter: 0 0 1 2 3 0 1 2 3
24.3.2 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 147. Counter timing diagram, internal clock divided by 1

Figure 148. Counter timing diagram, internal clock divided by 2
Figure 149. Counter timing diagram, internal clock divided by 4

Figure 150. Counter timing diagram, internal clock divided by N
Figure 151. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

Figure 152. Counter timing diagram, update event when ARPE=1 (TIMx_ARR 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.

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 153. Counter timing diagram, internal clock divided by 1**

- **CK_PSC**
- **CNT_EN**
- **Timer clock = CK_CNT**
- **Counter register**
- **Counter underflow (cnt_udf)**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

**Figure 154. Counter timing diagram, internal clock divided by 2**

- **CK_PSC**
- **CNT_EN**
- **Timer clock = CK_CNT**
- **Counter register**
- **Counter underflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
Figure 155. Counter timing diagram, internal clock divided by 4

Figure 156. 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 158. 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 24.4: TIM1 registers).
Figure 159. Counter timing diagram, internal clock divided by 2

CK_PSC

CNT_EN

Timer clock = CK_CNT

Counter register: 0003 0002 0001 0000 0001 0002 0003

Counter underflow

Update event (UEV)

Update interrupt flag (UIF)

Figure 160. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

CK_PSC

CNT_EN

Timer clock = CK_CNT

Counter register: 0034 0035 0036 0035

Counter overflow

Update event (UEV)

Update interrupt flag (UIF)

Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow
Figure 161. Counter timing diagram, internal clock divided by N

- **CK_PSC**: Timer clock
- **Timer clock = CK_CNT**: Internal clock
- **Counter register**: 20 1F 01 00
- **Counter underflow**:
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

Figure 162. Counter timing diagram, update event with ARPE=1 (counter underflow)

- **CK_PSC**: Timer clock
- **CEN**: Auto-reload preloaded
- **Timer clock = CK_CNT**: Internal clock
- **Counter register**:
- **Counter underflow**:
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
- **Auto-reload preloaded**
- **Write a new value in TIMx_ARR**
- **Auto-reload active**
24.3.3 Repetition counter

Section 24.3.1: 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 $2 \times T_{ck}$, 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 164). 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 underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 164. 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></td>
<td>Downcounting</td>
</tr>
<tr>
<td>TIMx_RCR = 1</td>
<td>Upcounting</td>
</tr>
<tr>
<td></td>
<td>Downcounting</td>
</tr>
<tr>
<td>TIMx_RCR = 2</td>
<td>Upcounting</td>
</tr>
<tr>
<td></td>
<td>Downcounting</td>
</tr>
<tr>
<td>TIMx_RCR = 3</td>
<td>Upcounting</td>
</tr>
<tr>
<td></td>
<td>Downcounting</td>
</tr>
<tr>
<td>TIMx_RCR = 3 and re-synchronization</td>
<td>Upcounting (by SW)</td>
</tr>
<tr>
<td></td>
<td>Downcounting (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.
24.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:
- external clock (external clock mode 2, see Section 24.3.5)
- trigger for the slave mode (see Section 24.3.26)
- PWM reset input for cycle-by-cycle current regulation (see Section 24.3.7)

Figure 165 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 165. External trigger input block](image)

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] and the TIM1_OR1[1:0] bitfields.

![Figure 166. TIM1 ETR input circuitry](image)
24.3.5 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 167* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

*Figure 167. Control circuit in normal mode, internal clock divided by 1*

---

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 170 gives an overview of the external trigger input block.

Figure 170. External trigger input block

1. Refer to Figure 166: 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 \( \text{ETF}[3:0]=0000 \) in the TIMx_SMCR register.
2. Set the prescaler by writing \( \text{ETPS}[1:0]=01 \) in the TIMx_SMCR register.
3. Select rising edge detection on the ETR pin by writing \( \text{ETP}=0 \) in the TIMx_SMCR register.
4. Enable external clock mode 2 by writing \( \text{ECE}=1 \) in the TIMx_SMCR register.
5. Enable the counter by writing \( \text{CEN}=1 \) 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 \( \frac{1}{4} \) of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

![Figure 171. Control circuit in external clock mode 2](image-url)
24.3.6 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 172 to Figure 175* 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 172. 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 173. Capture/compare channel 1 main circuit*
Figure 174. 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 175. 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.

### 24.3.7 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.

24.3.8 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_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 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 177. PWM input mode timing](image)

### 24.3.9 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.
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 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 178.
24.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated 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 669.

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 179 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 179. Edge-aligned PWM waveforms (ARR=8)

Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the Downcounting mode on page 673.

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 676.

Figure 180 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.

### 24.3.12 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 181** 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.
24.3.13 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 182 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.

![Figure 181. Generation of 2 phase-shifted PWM signals with 50% duty cycle](image-url)
24.3.14 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].
The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to Section 24.3.27: ADC synchronization for more details.

24.3.15 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 162: Output control bits for complementary OCx and OCxN channels with break feature on page 745 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 184. Complementary output with dead-time insertion**

**Figure 185. Dead-time waveforms with delay greater than the negative pulse**
Figure 186. 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 24.4.20: TIM1 break and dead-time register (TIM1_BDTR) 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 a specific waveform to be sent (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.

24.3.16 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 the outputs to be enabled/disabled 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 162: Output control bits for complementary OCx and OCxN channels with break feature on page 745 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_AF1 and TIMx_AF2 registers.

The sources for break (BRK) channel are:

An external source connected to one of the BKIN pin (as per selection done in the SYSCFG_CFRG2 register), with polarity selection and optional digital filtering

An internal source:
- the CPU1 Cortex®-M4 LOCKUP output
- the PVD output
- the SRAM parity error signal
- a flash memory ECC double error detection
- 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 SYSCFG_CFRG2 register), 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 187 below.

**Figure 187. 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 OSSI 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 OSSI=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 OSSI=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:** If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value.

If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

**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 the configuration of several parameters to be frozen (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 24.4.20: TIM1 break and dead-time register (TIM1_BDTR)*. The LOCK bits can be written only once after an MCU reset.

*Figure 188* shows an example of behavior of the outputs in response to a break.
Figure 188. Various output behavior in response to a break event on BRK (OSSI = 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 158.

**Note:** BRK2 must only be used with OSSR = OSSI = 1.

### Table 158. 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>
<th>OCxN output (low side switches)</th>
<th>OCx output (high side switches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>X</td>
<td>- Inactive then forced output state (after a deadtime)</td>
<td>ON after deadtime insertion</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Outputs disabled if OSSI = 0 (control taken over by GPIO logic)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inactive</td>
<td>Active</td>
<td>Inactive</td>
<td>OFF</td>
<td>OFF</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 189** 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 189. PWM output state following BRK and BRK2 pins assertion (OSSI=1)**
24.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on Figure 191.

They allow the following:

- 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, 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 is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents 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 159)
**Table 159. 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>

**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:

1. The BKDSRM (BK2DSRM) bit must be set to release the output control
2. 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)
3. 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.

**Figure 191. Output redirection (BRK2 request not represented)**
24.3.18 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 transition to the active state, on the following PWM cycle. 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 192 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.

![Figure 192. Clearing TIMx OCxREF](image)

**Note:** In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.
24.3.19 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 193 describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 193. 6-step generation, COM example (OSSR=1)
**24.3.20 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 194. Example of one pulse mode.](MS31099V1)

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 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 ($\text{TIMx\_ARR} - \text{TIMx\_CCR1}$).

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 Tlx 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.

### 24.3.21 Retriggerable one pulse mode

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 24.3.20:

- 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.
24.3.22 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 160. 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 used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The Figure 196 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_CCMR1 register, TI2FP2 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 160. 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></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>Counting on TI1 only</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td>Counting on TI2 only</td>
<td>High</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>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>No Count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No Count</td>
<td>No Count</td>
</tr>
</tbody>
</table>

**Figure 196. Example of counter operation in encoder interface mode.**
Figure 197 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

**Figure 197. 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).

### 24.3.23 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.
24.3.24 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 198 below.

![Figure 198. Measuring time interval between edges on 3 signals](image)

24.3.25 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) referred to as “interfacing timer” in Figure 199. 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 172: Capture/compare channel (example: channel 1 input stage) on page 687). 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 199 describes this example.
Figure 199. Example of Hall sensor interface

Interfacing timer

Counter (CNT)

CCR2

TRGO=OC2REF

CCI

TCM

OC1

OC1N

OC2

OC2N

OC3

OC3N

Write CCxE, CCxNE, and OCxM for next step
24.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 25.3.19: Timer synchronization for details. 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:

- 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).
- 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.
- 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 200. Control circuit in reset mode**
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 201. Control circuit in Gated mode](image_url)

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 202. Control circuit in trigger mode**

<table>
<thead>
<tr>
<th>TI2</th>
<th>cnt_en</th>
<th>Counter clock = ck_cnt = ck_psc</th>
<th>Counter register</th>
<th>TIF</th>
</tr>
</thead>
</table>

**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 203. 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.*
24.3.27 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 183 on page 699.

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.

24.3.28 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.

24.3.29 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIM1_STOP configuration bit in DBGMCU module.

For safety purposes, when the counter is stopped, 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 Debug support (DBG).
24.4 TIM1 registers

Refer to for a list of abbreviations used in register descriptions.

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

24.4.1 TIM1 control register 1 (TIM1_CR1)

Address offset: 0x00
Reset value: 0x0000

<table>
<thead>
<tr>
<th></th>
<th>UIFREMAP</th>
<th>CKD[1:0]</th>
<th>ARPE</th>
<th>CMS[1:0]</th>
<th>DIR</th>
<th>OPM</th>
<th>URS</th>
<th>UDIS</th>
<th>CEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:12 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>
</tr>
<tr>
<td>Bit 11 UIFREMAP: UIF status bit remapping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 10 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>
</tr>
<tr>
<td>Bits 9:8 CKD[1:0]: Clock division</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>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):</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00: tDTS=1/CK_INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01: tDTS=2/CK_INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10: tDTS=4/CK_INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11: Reserved, do not program this value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: tDTS = 1/fDTS, CK_INT = 1/fCK_INT.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 7 ARPE: Auto-reload preload enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: TIMx_ARR register is not buffered</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: TIMx_ARR register is buffered</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 6:5 CMS[1:0]: Center-aligned mode selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
24.4.2 TIM1 control register 2 (TIM1_CR2)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 32</th>
<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>
<td>rw</td>
</tr>
</tbody>
</table>

24.4.2 TIM1 control register 2 (TIM1_CR2)
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** - OC1REFC signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REFC rising or OC6REFC 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  **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1N bit

Bit 10  **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

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_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 selected 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 CC11IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC1REFC signal is used as trigger output (TRGO)

101: **Compare** - OC2REFC signal is used as trigger output (TRGO)

110: **Compare** - OC3REFC signal is used as trigger output (TRGO)

111: **Compare** - OC4REFC 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
24.4.3 TIM1 slave mode control register (TIM1_SMCR)

Address offset: 0x08
Reset value: 0x0000 0000

| 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 |
| 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 |
| 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: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111). 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). 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 f_{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 f_{DTS}
- 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

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 161: TIM1 internal trigger connection on page 731 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 the OCREF_CLR input
- 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 (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

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 161. 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>-</td>
<td>TIM2</td>
<td>-</td>
<td>TIM17 OC1</td>
</tr>
</tbody>
</table>

#### 24.4.4 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></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>
</tbody>
</table>

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
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  **COMDE**: COM DMA request enable
       0: COM DMA request disabled
       1: COM DMA request enabled

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   **BIE**: Break interrupt enable
       0: Break interrupt disabled
       1: Break interrupt enabled

Bit 6   **TIE**: Trigger interrupt enable
       0: Trigger interrupt disabled
       1: Trigger interrupt enabled

Bit 5   **COMIE**: COM interrupt enable
       0: COM interrupt disabled
       1: COM interrupt enabled

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
24.4.5 TIM1 status register (TIM1_SR)

Address offset: 0x10

<table>
<thead>
<tr>
<th>Bits 31:18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17</td>
<td>CC6IF: Compare 6 interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1IF description (Note: Channel 6 can only be configured as output)</td>
</tr>
<tr>
<td>Bit 16</td>
<td>CC5IF: Compare 5 interrupt flag</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1IF description (Note: Channel 5 can only be configured as output)</td>
</tr>
<tr>
<td>Bits 15:14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 13</td>
<td>SBIF: System Break interrupt flag</td>
</tr>
<tr>
<td></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.</td>
</tr>
<tr>
<td></td>
<td>This flag must be reset to re-start PWM operation.</td>
</tr>
<tr>
<td></td>
<td>0: No break event occurred.</td>
</tr>
<tr>
<td></td>
<td>1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.</td>
</tr>
<tr>
<td>Bit 12</td>
<td>CC4OF: Capture/Compare 4 overcapture flag</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>Bit 11</td>
<td>CC3OF: Capture/Compare 3 overcapture flag</td>
</tr>
<tr>
<td></td>
<td>Refer to CC1OF description</td>
</tr>
<tr>
<td>Bit 10</td>
<td>CC2OF: Capture/Compare 2 overcapture flag</td>
</tr>
<tr>
<td></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 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  **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
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).
24.4.6 TIM1 event generation register (TIM1_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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B2G</td>
<td></td>
<td>BG</td>
<td></td>
<td>TG</td>
<td></td>
<td>COMG</td>
<td></td>
<td>CC4G</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></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 CCxE, CCxNE and OCxM bits to be updated.

**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
Advanced-control timer (TIM1) RM0434

24.4.7 TIM1 capture/compare mode register 1 (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>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 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 IC1PSC[1:0] description.
24.4.8 **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>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>
</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
Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode
Refer to OC1M[3:0] 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 (OCREF_CLR input or 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 can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

- **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 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:** 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:** 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.

**Note:** The OC1M[3] bit is not contiguous, located in bit 16.
24.4.9 TIM1 capture/compare mode register 2 (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:
24.4.10 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>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  rw  rw  rw  rw  rw  rw  rw  rw
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
        Refer to OC1CE description.

Bits 24, 14:12  **OC4M[3:0]**: Output compare 4 mode
        Refer to OC3M[3:0] 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.

Bit 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 16, 6:4  **OC3M[3:0]**: Output compare 3 mode
        Refer to OC1M[3:0] 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).**
### 24.4.11 TIM1 capture/compare enable register (TIM1_CCER)

Address offset: 0x20

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>CC6P</th>
<th></th>
<th>CC6E</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></td>
</tr>
</tbody>
</table>

Bits 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

Bits 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, 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.  

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  

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=0:** 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:** The configuration is reserved, it must not be used.  

*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

0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to Table 162 for details.

**Note**: 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 162. Output control bits for complementary OCx and OCxN channels with break feature |
|----|----|----|----|---|---|---|---|
| MOE bit | OSSI bit | OSSR bit | CCxE bit | CCxNE bit | OCx output state | OCxN output state |
| 1 | X | X | 0 | 0 | Output disabled (not driven by the timer: Hi-Z) | OCx=0, OCxN=0 |
| 0 | 0 | 1 | Output disabled (not driven by the timer: Hi-Z) | OCx=0 | OCxREF + Polarity | OCxN = OCxREF xor CCxNP |
| 0 | 1 | 0 | OCxREF + Polarity | OCx=CCxP | Output Disabled (not driven by the timer: Hi-Z) | OCxN=0 |
| X | 1 | 1 | OCREF + Polarity + dead-time | Complementary to OCREF (not OCREF) + Polarity + dead-time |
| 1 | 0 | 1 | Off-State (output enabled with inactive state) | OCx=CCxP | OCxREF + Polarity | OCxN = OCxREF x or CCxNP |
| 1 | 1 | 0 | OCxREF + Polarity | OCx=OCxREF xor CCxP | Off-State (output enabled with inactive state) | OCxN=CCxNP |
| 0 | X | X | X | X | Output disabled (not driven by the timer: Hi-Z). |
| 0 | 0 | 0 | Off-State (output enabled with inactive state) |
| 1 | 0 | 1 | Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). |
| 1 | 1 | 1 | Then (this is valid only if BRK is triggered), 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 (may cause a short circuit when driving switches in half-bridge configuration). |

**Note**: BRK2 can only be used if OSSI = OSSR = 1.

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**: 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.
## 24.4.12 TIM1 counter (TIM1_CNT)

Address offset: 0x24  
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>r</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>
</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>

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 the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

## 24.4.13 TIM1 prescaler (TIM1_PSC)

Address offset: 0x28  
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>PSC[15:0]</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>
<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 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”).

## 24.4.14 TIM1 auto-reload register (TIM1_ARR)

Address offset: 0x2C  
Reset value: 0xFFFF

<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>ARR[15:0]</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>
<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 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 24.3.1: Time-base unit on page 667 for more details about ARR update and behavior.  
The counter is blocked while the auto-reload value is null.
24.4.15 TIM1 repetition counter register (TIM1_RCR)

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>
<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 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 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
- the number of half PWM period in center-aligned mode.

24.4.16 TIM1 capture/compare register 1 (TIM1_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:** 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.
24.4.17  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 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.

24.4.18  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.
24.4.19  TIM1 capture/compare register 4
(TIM1_CCR4)

Address offset: 0x40
Reset value: 0x0000

### TIM1_CCR4[15:0]

- **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.

24.4.20  TIM1 break and dead-time register
(TIM1_BDTR)

Address offset: 0x44
Reset value: 0x0000 0000

### TIM1_BDTR[31:0]

- **Reserved**: Bits 31:30
- **BK2BID**: Break2 bidirectional
- **BKBID**: Refer to BKBID description
- **BK2DSRM**: BK2DSRM
- **BDSTM**: BKDSRM
- **BK2P**: BK2P
- **BK2E**: BK2E
- **BK2F[3:0]**: BK2F[3:0]
- **BKF[3:0]**: BKF[3:0]
- **MOE**: MOE
- **AOE**: AOE
- **BKP**: BKP
- **BKE**: BKE
- **OSSR**: OSSR
- **OSSI**: OSSI
- **LOCK[1:0]**: LOCK[1:0]
- **DTG[7:0]**: DTG[7:0]

**Note:** As the bits BK2BID, BK2BID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, 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  
*Note: The 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:

<table>
<thead>
<tr>
<th>Bit Pattern</th>
<th>Sampling Frequency</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>No filter, BRK2 acts asynchronously</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>f_{SAMPLING} = f_{CK_INT}</td>
<td>2</td>
</tr>
<tr>
<td>0010</td>
<td>f_{SAMPLING} = f_{CK_INT}</td>
<td>4</td>
</tr>
<tr>
<td>0011</td>
<td>f_{SAMPLING} = f_{CK_INT}</td>
<td>8</td>
</tr>
<tr>
<td>0100</td>
<td>f_{SAMPLING} = f_{DTS}/2</td>
<td>6</td>
</tr>
<tr>
<td>0101</td>
<td>f_{SAMPLING} = f_{DTS}/2</td>
<td>8</td>
</tr>
<tr>
<td>0110</td>
<td>f_{SAMPLING} = f_{DTS}/4</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>f_{SAMPLING} = f_{DTS}/4</td>
<td>8</td>
</tr>
<tr>
<td>1000</td>
<td>f_{SAMPLING} = f_{DTS}/8</td>
<td>6</td>
</tr>
<tr>
<td>1001</td>
<td>f_{SAMPLING} = f_{DTS}/8</td>
<td>8</td>
</tr>
<tr>
<td>1010</td>
<td>f_{SAMPLING} = f_{DTS}/16</td>
<td>5</td>
</tr>
<tr>
<td>1011</td>
<td>f_{SAMPLING} = f_{DTS}/16</td>
<td>6</td>
</tr>
<tr>
<td>1100</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>5</td>
</tr>
<tr>
<td>1101</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>6</td>
</tr>
<tr>
<td>1110</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>8</td>
</tr>
<tr>
<td>1111</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>8</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Bit Pattern</th>
<th>Sampling Frequency</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>No filter, BRK acts asynchronously</td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>f_{SAMPLING} = f_{CK_INT}</td>
<td>2</td>
</tr>
<tr>
<td>0010</td>
<td>f_{SAMPLING} = f_{CK_INT}</td>
<td>4</td>
</tr>
<tr>
<td>0011</td>
<td>f_{SAMPLING} = f_{CK_INT}</td>
<td>8</td>
</tr>
<tr>
<td>0100</td>
<td>f_{SAMPLING} = f_{DTS}/2</td>
<td>6</td>
</tr>
<tr>
<td>0101</td>
<td>f_{SAMPLING} = f_{DTS}/2</td>
<td>8</td>
</tr>
<tr>
<td>0110</td>
<td>f_{SAMPLING} = f_{DTS}/4</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>f_{SAMPLING} = f_{DTS}/4</td>
<td>8</td>
</tr>
<tr>
<td>1000</td>
<td>f_{SAMPLING} = f_{DTS}/8</td>
<td>6</td>
</tr>
<tr>
<td>1001</td>
<td>f_{SAMPLING} = f_{DTS}/8</td>
<td>8</td>
</tr>
<tr>
<td>1010</td>
<td>f_{SAMPLING} = f_{DTS}/16</td>
<td>5</td>
</tr>
<tr>
<td>1011</td>
<td>f_{SAMPLING} = f_{DTS}/16</td>
<td>6</td>
</tr>
<tr>
<td>1100</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>5</td>
</tr>
<tr>
<td>1101</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>6</td>
</tr>
<tr>
<td>1110</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>8</td>
</tr>
<tr>
<td>1111</td>
<td>f_{SAMPLING} = f_{DTS}/32</td>
<td>8</td>
</tr>
</tbody>
</table>

**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 OSS1 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 24.4.11: 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 BRK sources, as per **Figure 187: 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 24.4.11: 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).
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 24.4.11: TIM1 capture/compare enable register (TIM1_CCR1)).

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 BK2BID, BK2BID, BK2DSRM, BK2DSRM, BK2P, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, 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_CCR1 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 tDTG with tDTG = tDTS.

DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x tDTG with tDTG = 2 x tDTS.

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x tDTG with tDTG = 8 x tDTS.

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x tDTG with tDTG = 16 x tDTS.

Example if tDTS = 125 ns (8 MHz), 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).

### 24.4.21 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>
<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 15:13 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).

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>1 transfer</td>
</tr>
<tr>
<td>00001</td>
<td>2 transfers</td>
</tr>
<tr>
<td>00010</td>
<td>3 transfers</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10001</td>
<td>18 transfers</td>
</tr>
</tbody>
</table>

**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:
  \[(\text{TIMx}\_\text{CR1 address}) + \text{DBA} + (\text{DMA index})\], where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is 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 is transferred to each of the 7 registers.
- If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains 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:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>TIMx_CR1</td>
</tr>
<tr>
<td>00001</td>
<td>TIMx_CR2</td>
</tr>
<tr>
<td>00010</td>
<td>TIMx_SMCR</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### 24.4.22 TIM1 DMA address for full transfer (TIM1_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

<table>
<thead>
<tr>
<th>Offset</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>rw</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>

ST
24.4.23 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>

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TI1_RMP**: Input Capture 1 remap
0: TIM1 input capture 1 is connected to I/O
1: TIM1 input capture 1 is connected to COMP1 output

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **TIM1_ETR_ADC_RMP[1:0]**: TIM1_ETR_ADC remapping capability
00: TIM1_ETR is not connected to ADC AWDX (must be selected when the ETR comes from the ETR input pin)
01: TIM1_ETR is connected to ADC AWDX1
10: TIM1_ETR is connected to ADC AWDX2
11: TIM1_ETR is connected to ADC AWDX3

*Note: ADC AWDX sources are 'ORed' with the TIM1_ETR input signals. When ADC AWDX is used, it is necessary to make sure that the corresponding TIM1_ETR input pin is not enabled in the alternate function controller.*

24.4.24 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:
### Advanced-control timer (TIM1) RM0434

#### 24.4.25 TIM1 capture/compare register 5 (TIM1_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

| Bit 31:25 | Reserved, must be kept at reset value. |
| Bit 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. |

### 24.4.25 TIM1 capture/compare register 5 (TIM1_CCR5)

| Bit 31:25 | Reserved, must be kept at reset value. |
| Bit 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. |
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 TIMxCCMR2).
*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 TIMxCCMR1).
*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 OC1REFC5
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 TIMxCCMR1).
*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.

24.4.26 TIM1 capture/compare register 6
(TIM1_CCR6)
Address offset: 0x5C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>CCR6[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw 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.
### 24.4.27 TIM1 alternate function option register 1 (TIM1_AF1)

**Address offset:** 0x60  
**Reset value:** 0x0000 0001

| Bit 31-18 | Reserved, must be kept at reset value. |
| Bit 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  
Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register). |
| Bit 13-12 | 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 polarity is not inverted (active low if BKP=0, active high if BKP=1)  
1: COMP2 input polarity is inverted (active high if BKP=0, active low if BKP=1)  
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 polarity is not inverted (active low if BKP=0, active high if BKP=1)  
1: COMP1 input polarity is inverted (active high if BKP=0, active low if BKP=1)  
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 polarity is not inverted (active low if BKP=0, active high if BKP=1)  
1: BKin input polarity is inverted (active high if BKP=0, active low if BKP=1)  
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. |
24.4.28 TIM1 Alternate function register 2 (TIM1_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

| Bit 31:12 |          |          |          |          |          |          |          |          |          |          |          |          |          |          |          |
|-----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 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 polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
1: COMP2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

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 polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
1: COMP1 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)
*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 polarity is not inverted (active low if BK2P=0, active high if BK2P=1)
1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)
*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 187: Break and Break2 circuitry overview.
24.4.29 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></td>
<td></td>
<td></td>
<td></td>
<td></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>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<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 27:24 \textbf{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 \textbf{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 \textbf{TI2SEL[3:0]}: selects TI2[0] to TI2[15] input
0000: TIM1_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 \textbf{TI1SEL[3:0]}: selects TI1[0] to TI1[15] input
0000: TIM1_CH1 input
Others: Reserved
### 24.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</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>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIM1_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x04</td>
<td>TIM1_CCMR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x08</td>
<td>TIM1_SMCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x0C</td>
<td>TIM1_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x10</td>
<td>TIM1_SRR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x14</td>
<td>TIM1_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x18</td>
<td>TIM1_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Output/Compare mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x1C</td>
<td>TIM1_CCMR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Output/Compare mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
<tr>
<td>0x20</td>
<td>TIM1_CCER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
</tr>
</tbody>
</table>

**Table 163. TIM1 register map and reset values**

*Note: The table above shows the hexadecimal values for the registers.*
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Name</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>TIM1_CNT</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>CNT[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0x04</td>
<td>0x24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>0x08</td>
<td>0x28</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>0x12</td>
<td>0x30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>0x16</td>
<td>0x34</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>0x20</td>
<td>0x38</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>0x24</td>
<td>0x40</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>0x28</td>
<td>0x44</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>0x32</td>
<td>0x48</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>36</td>
<td>0x36</td>
<td>0x50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TIM1_PSC</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>PSC[15:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_ARR</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>ARR[15:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_RCR</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>REP[15:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_CCR1</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>CCR1[15:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_CCR2</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>CCR2[15:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_CCR3</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>CCR3[15:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_CCR4</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>CCR4[15:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_BDTR</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>LOC[1:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_DCR</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>DBL[4:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_DMAR</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>DMAB[31:0]</td>
</tr>
<tr>
<td></td>
<td>TIM1_OR1</td>
<td>0</td>
<td>0x00</td>
<td>16</td>
<td>T1_RMP</td>
</tr>
</tbody>
</table>

Reset values:
- TIM1_CNT: 0x0000
- TIM1_PSC: 0x0000
- TIM1_ARR: 0x0000
- TIM1_RCR: 0x0000
- TIM1_CCR1: 0x0000
- TIM1_CCR2: 0x0000
- TIM1_CCR3: 0x0000
- TIM1_CCR4: 0x0000
- TIM1_BDTR: 0x0000
- TIM1_DCR: 0x0000
- TIM1_DMAR: 0x0000
- TIM1_OR1: 0x0000

Table 163. TIM1 register map and reset values (continued)
Refer to Section 2.2 on page 66 for the register boundary addresses.
25 General-purpose timer (TIM2)

25.1 TIM2 introduction

The general-purpose timer TIM2 consists of a 32-bit auto-reload counter driven by a programmable prescaler.

The timer 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.

25.2 TIM2 main features

- 32-bit 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 204. General-purpose timer block diagram

Notes:
- Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA output

MSv40927V4
25.3 **TIM2 functional description**

25.3.1 **Time-base unit**

The main block of the programmable timer is a 32-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)

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 205* and *Figure 206* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 205. Counter timing diagram with prescaler division change from 1 to 2

- CK_PSC
- CEN
- Timer clock = CK_CNT
- Counter register: F7, F8, F9, FA, FB, FC, 00, 01, 02, 03
- Update event (UEV)
- Prescaler control register: 0, 1
- Prescaler buffer: 0, 1
- Prescaler counter: 0, 0, 1, 0, 1, 0, 1

Figure 206. Counter timing diagram with prescaler division change from 1 to 4

- CK_PSC
- CEN
- Timer clock = CK_CNT
- Counter register: F7, F8, F9, FA, FB, FC, 00, 01
- Update event (UEV)
- Prescaler control register: 0, 1, 3
- Prescaler buffer: 0, 3
- Prescaler counter: 0, 0, 1, 2, 3, 0, 1, 2, 3
25.3.2 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 207. Counter timing diagram, internal clock divided by 1

![Diagram showing counter timing diagram](MS3107BV2)
Figure 208. Counter timing diagram, internal clock divided by 2

Figure 209. Counter timing diagram, internal clock divided by 4
**Figure 210. Counter timing diagram, internal clock divided by N**

CK_PSC

Timer clock = CK_CNT

Counter register

Counter overflow

Update event (UEV)

Update interrupt flag (UIF)

**Figure 211. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)**

CK_PSC

CEN

Timer clock = CK_CNT

Counter register

Counter overflow

Update event (UEV)

Update interrupt flag (UIF)

Auto-reload preload register

Write a new value in TIMx_ARR
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 213. Counter timing diagram, internal clock divided by 1**

**Figure 214. Counter timing diagram, internal clock divided by 2**
Figure 215. Counter timing diagram, internal clock divided by 4

![Diagram of counter timing with CK_PSC, CNT_EN, Timer clock = CK_CNT, Counter register (0001 0000 0000 0001), Counter underflow, Update event (UEV), Update interrupt flag (UIF)].

Figure 216. Counter timing diagram, internal clock divided by N

![Diagram of counter timing with 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 218. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6**

<table>
<thead>
<tr>
<th>CK_PSC</th>
<th>CEN</th>
<th>Timerclock = CK_CNT</th>
<th>Counter register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>04 03 02 01 00 01 02 03 04 05 06 05 04 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Counter underflow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Counter overflow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update event (UEV)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Update interrupt flag (UIF)</td>
</tr>
</tbody>
</table>

1. Here, center-aligned mode 1 is used (for more details refer to Section 25.4.1: TIM2 control register 1 (TIM2_CR1) on page 810).
Figure 219. Counter timing diagram, internal clock divided by 2

Figure 220. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

1. Center-aligned mode 2 or 3 is used with an UIF on overflow.
Figure 221. 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 222. 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
- Write a new value in TIMx_ARR
- Auto-reload active register
25.3.3 Clock selection

The counter clock can be provided by the following clock sources:
- Internal clock (CK_INT)
- External clock mode1: external input pin (Tlx)
- 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 804 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 224 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.

Figure 224. Control circuit in normal mode, internal clock divided by 1

Figure 225. TI2 external clock connection example

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.

![Control circuit in external clock mode 1](MS31087V2)

**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 227* gives an overview of the external trigger input block.
1. As per ETR_RMP bit programming.

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 and the ETR_RMP bit in the TIM2_OR1 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.
25.3.4 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 T1x input to generate a filtered signal T1xF. Then, an edge detector with polarity selection generates a signal (T1xFPx) 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.
25.3.5 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 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 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.
25.3.6 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 Tlx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TlxFP 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 T11 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper T11x 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 232. 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 T11FP1 and T12FP2 are connected to the slave mode controller.
25.3.7 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.

25.3.8 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 CCxCE 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 CCxCE=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 233.

**Figure 233. Output compare mode, toggle on OC1**

### 25.3.9 PWM mode

Pulse width modulation mode permits 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_CCRER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCRER register. Refer to the TIMx_CCRERx 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 769](#).

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 234 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

**Figure 234. 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 772*.

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 775*.

*Figure 235* 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.
25.3.10 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 236* shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

25.3.11 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 237 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

25.3.12 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 transition to the active state, on the following PWM cycle. 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 transition to the active state, on the following PWM cycle.

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 238 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 238. Clearing TIMx OCxREF](image)

Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.
25.3.13 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:

\[ \text{CNT}<\text{CCR}_x \leq \text{ARR} \text{ (in particular, } 0<\text{CCR}_x) \]

**Figure 239. 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 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 ($\text{TIMx\_ARR} - \text{TIMx\_CCR1}$).

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 Tlx 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.

**25.3.14 Retriggerable one pulse mode**

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 25.3.13:

- 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.
25.3.15 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 164. 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 indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

**Figure 241** 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_CCMR1 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 241. Example of counter operation in encoder interface mode**

**Table 164. Counting direction versus encoder signals**

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (TI1FP1 for T12, TI2FP2 for TI1)</th>
<th>TI1FP1 signal</th>
<th>TI2FP2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>Counting on T11 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>Counting on T12 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 T11 and T12</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>

**Figure 242** 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.

25.3.16 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 permits 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).

25.3.17 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 24.3.25: Interfacing with Hall sensors on page 716.
25.3.18 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_CCRER 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 can be sent if enabled (depending on the TIE bit 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 243. 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:
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_CCMR1 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 244. Control circuit in gated mode**

![Control Circuit in Gated Mode](MS31402V1)

### Note:

The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

### 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 245. Control circuit in trigger mode**

![Control Circuit in Trigger Mode](image)

**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 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 246. Control circuit in external clock mode 2 + trigger mode

25.3.19 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 247: Master/Slave timer example and Figure 248: Master/slave connection example with 1 channel only timers present an overview of the trigger selection and the master mode selection blocks.
Note: The timers with one channel only (see Figure 248) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave. The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination’s timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM1 can be configured to act as a prescaler for TIM2. Refer to Figure 247. To do this:

![Figure 247. Master/Slave timer example](image1)

![Figure 248. Master/slave connection example with 1 channel only timers](image2)
1. Configure TIM1 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM1_CR2 register, a rising edge is output on TRGO each time an update event is generated.

2. To connect the TRGO output of TIM1 to TIM2, TIM2 must be configured in slave mode using ITR0 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=0000).

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 TIM1 trigger signal (which correspond to the TIM1 counter overflow).

4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

**Note:** If OCx is selected on TIM1 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 1. Refer to Figure 247 for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM1 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 TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).

2. Configure the TIM1 OC1REF waveform (TIM1_CCMR1 register).

3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 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 (TIM_CR1 register).

6. Start TIM by writing '1 in the CEN bit (TIM1_CR1 register).

**Note:** The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

![Figure 249. Gating TIM2 with OC1REF of TIM1](MS32695V2)

In the example in Figure 249, 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 TIM1. 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 250), we synchronize TIM1 and TIM2. TIM1 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 TIM1 is disabled by writing ‘0 to the CEN bit in the TIM1_CR1 register:

1. Configure TIM1 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM1_CR2 register).
2. Configure the TIM1 OC1REF waveform (TIM1_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM1 by writing ‘1 in UG bit (TIM1_EGR register).
6. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0x7E’ in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
9. Start TIM1 by writing ‘1 in the CEN bit (TIM1_CR1 register).
10. Stop TIM1 by writing ‘0 in the CEN bit (TIM1_CR1 register).

**Figure 250. Gating TIM2 with Enable of TIM1**

Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to Figure 247 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 \( (f_{CK\_INT} = f_{CK\_INT}/3) \).
1. Configure TIM1 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM1_CR2 register).
2. Configure the TIM1 period (TIM1_ARR registers).
3. Configure TIM2 to get the input trigger from TIM1 (TS=00000 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM1 by writing '1 in the CEN bit (TIM1_CR1 register).

**Figure 251. Triggering TIM2 with update of TIM1**

As in the previous example, both counters can be initialized before starting counting. **Figure 252** shows the behavior with the same configuration as in **Figure 251** but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

**Figure 252. Triggering TIM2 with Enable of TIM1**

*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO signal 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.*
25.3.20 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.
25.3.21 **Debug mode**

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIM2_STOP configuration bit in DBGMCU module. For more details, refer to [Section 41.8.3: DBGMCU CPU1 APB1 peripheral freeze register 1 (DBGMCU_APB1FZR1)](#).
25.4 **TIM2 registers**

In this section, “TIMx” should be understood as “TIM2” since there is only one instance of this type of timer for the products to which this reference manual applies.

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).

### 25.4.1 TIM2 control register 1 (TIM2_CR1)

**Address offset**: 0x00  
**Reset value**: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

**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 (ETR, TIx),
- 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 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)*
25.4.2 TIM2 control register 2 (TIM2_CR2)

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>
</tbody>
</table>

**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.
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 24.3.25: Interfacing with Hall sensors on page 716

Bits 6:4  **MMS[2:0]**: Master mode selection

These bits permit 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.

100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC 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.
25.4.3 TIM2 slave mode control register (TIM2_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>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 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\(^\uparrow\) 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:* Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).

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).

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 \( \text{N} \) consecutive events are needed to validate a transition on the output:

- **0000**: No filter, sampling is done at \( f_{\text{DTS}} \)
- **0001**: \( f_{\text{SAMPLING}} = f_{\text{CK\_INT}}, \text{N}=2 \)
- **0010**: \( f_{\text{SAMPLING}} = f_{\text{CK\_INT}}, \text{N}=4 \)
- **0011**: \( f_{\text{SAMPLING}} = f_{\text{CK\_INT}}, \text{N}=8 \)
- **0100**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/2, \text{N}=6 \)
- **0101**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/2, \text{N}=8 \)
- **0110**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/4, \text{N}=6 \)
- **0111**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/4, \text{N}=8 \)
- **1000**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/8, \text{N}=6 \)
- **1001**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/8, \text{N}=8 \)
- **1010**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/16, \text{N}=5 \)
- **1011**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/16, \text{N}=6 \)
- **1100**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/16, \text{N}=8 \)
- **1101**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/32, \text{N}=5 \)
- **1110**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/32, \text{N}=6 \)
- **1111**: \( f_{\text{SAMPLING}} = f_{\text{DTS}}/32, \text{N}=8 \)

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)
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 165: TIM2 internal trigger connection on page 816 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 the OCREF_CLR input
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 signal 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 165. TIM2 internal trigger connection

<table>
<thead>
<tr>
<th>Slave TIM</th>
<th>ITR0</th>
<th>ITR1</th>
<th>ITR2 - ITR8</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM2</td>
<td></td>
<td></td>
<td>USB_FS_SOF(1)</td>
</tr>
</tbody>
</table>

1. This connection is valid only when the ITR_RMP bit is set in the TIM2 OR1 register.

## 25.4.4 TIM2 DMA/Interrupt enable register (TIM2_DIER)

Address offset: 0x0C

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</th>
<th>Bits</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>
<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>
</tr>
</tbody>
</table>

**Bits 15:14** Reserved, must be kept at reset value.

**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.

### 25.4.5 TIM2 status register (TIM2_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></td>
<td></td>
<td></td>
<td></td>
<td></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>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>

---

RM0434 Rev 13

817/1530
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
**TIM2 event generation register (TIM2_EGR)**

Address offset: 0x14

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>TG: Trigger generation</td>
</tr>
<tr>
<td>13</td>
<td>This bit is set by software in order to generate an event, it is automatically cleared by hardware.</td>
</tr>
<tr>
<td>12</td>
<td>0: No action</td>
</tr>
<tr>
<td>11</td>
<td>1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>CC4G: Capture/compare 4 generation</td>
</tr>
<tr>
<td>8</td>
<td>Refer to CC1G description</td>
</tr>
<tr>
<td>7</td>
<td>CC3G: Capture/compare 3 generation</td>
</tr>
<tr>
<td>6</td>
<td>Refer to CC1G description</td>
</tr>
<tr>
<td>5</td>
<td>CC2G: Capture/compare 2 generation</td>
</tr>
<tr>
<td>4</td>
<td>Refer to CC1G description</td>
</tr>
<tr>
<td>3</td>
<td>CC1G: Capture/compare 1 generation</td>
</tr>
<tr>
<td>2</td>
<td>UG: Update generation</td>
</tr>
<tr>
<td>1</td>
<td>This bit is set by hardware on an update event. It is cleared by software.</td>
</tr>
<tr>
<td>0</td>
<td>0: No update occurred</td>
</tr>
<tr>
<td></td>
<td>1: Update interrupt pending. This bit is set by hardware when the registers are updated:</td>
</tr>
<tr>
<td></td>
<td>At overflow or underflow and if UDIS=0 in the TIMx_CR1 register.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**25.4.6 TIM2 event generation register (TIM2_EGR)**

Address offset: 0x14

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>TG: Trigger generation</td>
</tr>
<tr>
<td>13</td>
<td>This bit is set by software in order to generate an event, it is automatically cleared by hardware.</td>
</tr>
<tr>
<td>12</td>
<td>0: No action</td>
</tr>
<tr>
<td>11</td>
<td>1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>CC4G: Capture/compare 4 generation</td>
</tr>
<tr>
<td>8</td>
<td>Refer to CC1G description</td>
</tr>
<tr>
<td>7</td>
<td>CC3G: Capture/compare 3 generation</td>
</tr>
<tr>
<td>6</td>
<td>Refer to CC1G description</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).

### 25.4.7 TIM2 capture/compare mode register 1 (TIM2_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:**

<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>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 31:16  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 \( f_{DTS} \)
- 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: \( f_{SAMPLING} = f_{DTS}/32, N=8 \)

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).
### 25.4.8 TIM2 capture/compare mode register 1 [alternate] (TIM2_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:

| Bit 31:25 | Reserved, must be kept at reset value. |
| Bit 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 |
| 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_CCER). |
| 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 |

---

| 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:** |

| Bit 31:25 | Reserved, must be kept at reset value. |
| Bit 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 |
| 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_CCER). |
| 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 can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

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 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: 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:** 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:** The OC1M[3] bit is not contiguous, located in bit 16.
General-purpose timer (TIM2) RM0434

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.

Bit 2  **OC1FE**: Output compare 1 fast enable
This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
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).

25.4.9  **TIM2 capture/compare mode register 2 (TIM2_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 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>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 31:16  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
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**

Bits 3:2 **IC3PSC[1:0]: Input capture 3 prescaler**

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).*

### 25.4.10 TIM2 capture/compare mode register 2 [alternate] (TIM2_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 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>

<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 31:25: 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
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

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable
Bit 2 **OC3FE**: Output compare 3 fast enable

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).**

### 25.4.11 TIM2 capture/compare enable register (TIM2_CCER)

Address offset: 0x20
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>CC4NP: Capture/Compare 4 output Polarity. Refer to CC1NP description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 13</td>
<td>CC4P: Capture/Compare 4 output Polarity. Refer to CC1P description</td>
</tr>
<tr>
<td>Bit 12</td>
<td>CC4E: Capture/Compare 4 output enable. Refer to CC1E description</td>
</tr>
<tr>
<td>Bit 11</td>
<td>CC3NP: Capture/Compare 3 output Polarity. Refer to CC1NP description</td>
</tr>
<tr>
<td>Bit 10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9</td>
<td>CC3P: Capture/Compare 3 output Polarity. Refer to CC1P description</td>
</tr>
<tr>
<td>Bit 8</td>
<td>CC3E: Capture/Compare 3 output enable. Refer to CC1E description</td>
</tr>
</tbody>
</table>
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 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).
- CC1NP=0, CC1P=1: 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).
- CC1NP=1, CC1P=1: 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.
- 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 (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td>1</td>
<td>Output enabled (tim_ocx = tim_ocxref + Polarity)</td>
</tr>
</tbody>
</table>

**Note:** The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO control and alternate function registers.

### 25.4.12 TIM2 counter (TIM2_CNT)

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
Address offset: 0x24
Reset value: 0x0000 0000

25.4.13 TIM2 counter [alternate] (TIM2_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:
- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24
Reset value: 0x0000 0000

Bits 31:0 CNT[31:0]: counter value

25.4.14 TIM2 prescaler (TIM2_PSC)

Address offset: 0x28
Reset value: 0x0000

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").
25.4.15 TIM2 auto-reload register (TIM2_ARR)

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

ARR[31:16]

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

Bits 31:0 **ARR[31:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the Section 25.3.1: Time-base unit on page 767 for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

25.4.16 TIM2 capture/compare register 1 (TIM2_CCR1)

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

CCR1[31:16]

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 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>

CCR1[15:0]

Bits 31:0 **CCR1[31:0]**: 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). The TIMx_CCR1 register is read-only and cannot be programmed.

25.4.17 TIM2 capture/compare register 2 (TIM2_CCR2)

Address offset: 0x38
Reset value: 0x0000 0000
25.4.18 TIM2 capture/compare register 3 (TIM2_CCR3)

Address offset: 0x3C
Reset value: 0x0000 0000

Bits 31:0 CCR3[31:0]: 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 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.

25.4.19 TIM2 capture/compare register 4 (TIM2_CCR4)

Address offset: 0x40
Reset value: 0x0000 0000
25.4.20 TIM2 DMA control register (TIM2_DCR)

Address offset: 0x48

Reset value: 0x0000

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.
25.4.21 TIM2 DMA address for full transfer (TIM2_DMAR)

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 | rw | rw | 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
(TIMx_CR1 address) + (DBA + DMA index) x 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).

25.4.22 TIM2 option register 1 (TIM2_OR1)

Address offset: 0x50
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 |
| rw | rw | rw | rw |

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

Bits 3:2 TIM4_RMP: Timer input 4 remap
Set and cleared by software.
00: TIM2 TI4 is connected to GPIO: Refer to Alternate Function mapping
01: TIM2 TI4 is connected to COMP1_OUT
10: TIM2 TI4 is connected to COMP2_OUT
11: TIM2 TI4 is connected to a logical OR between COMP1_OUT and COMP2_OUT

Bit 1 ETR_RMP: External trigger 1 remap
Set and cleared by software.
0: TIM2 ETR is connected to GPIO: Refer to Alternate Function mapping
1: LSE internal clock is connected to TIM2_ETR input

Bit 0 ITR_RMP: Internal trigger remap
0: TIM2 Internal trigger ITR2 is not connected
1: TIM2 Internal trigger ITR2 is connected to USB_FS_SOF

25.4.23 TIM2 alternate function option register 1 (TIM2_AF1)

Address offset: 0x60
Reset value: 0x0000 0000
### 25.4.24 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x68
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:12</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 11:8</td>
<td><strong>TI2SEL[3:0]</strong>: TI2[0] to TI2[15] input selection</td>
</tr>
<tr>
<td></td>
<td>These bits select the TI2[0] to TI2[15] input source.</td>
</tr>
<tr>
<td></td>
<td>0000: TIM2_CH2 input</td>
</tr>
<tr>
<td></td>
<td>Others: Reserved</td>
</tr>
<tr>
<td>Bit 7:4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 3:0</td>
<td><strong>TH1SEL[3:0]</strong>: TI1[0] to TI1[15] input selection</td>
</tr>
<tr>
<td></td>
<td>These bits select the TI1[0] to TI1[15] input source.</td>
</tr>
<tr>
<td></td>
<td>0000: TIM2_CH1 input</td>
</tr>
<tr>
<td></td>
<td>Others: Reserved</td>
</tr>
</tbody>
</table>
## 25.4.25 TIMx register map

TIMx registers are mapped as described in the table below:

### Table 167. TIM2 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>0x00</th>
<th>0x04</th>
<th>0x08</th>
<th>0x0C</th>
<th>0x10</th>
<th>0x14</th>
<th>0x18</th>
<th>0x1C</th>
<th>0x20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<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>
</tr>
<tr>
<td></td>
<td></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>
</tr>
<tr>
<td></td>
<td></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>
</tr>
<tr>
<td></td>
<td></td>
<td>1</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>TIMx_CR1</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
</tr>
<tr>
<td></td>
<td>TIMx_CR2</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
</tr>
<tr>
<td></td>
<td>TIMx_SMCR</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
</tr>
<tr>
<td></td>
<td>TIMx_DIER</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
</tr>
<tr>
<td></td>
<td>TIMx_5R</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
</tr>
<tr>
<td></td>
<td>TIMx_EGR</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
</tr>
<tr>
<td></td>
<td>TIMx_CCMR1</td>
<td>OC2M[3]</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
<td>Reserve</td>
</tr>
<tr>
<td></td>
<td>TIMx_CCER</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
</tr>
<tr>
<td></td>
<td>TIMx_CCER</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
<td>CCAP</td>
</tr>
</tbody>
</table>
### Table 167. TIM2 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset (Bits)</th>
<th>Type</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>TIMx_CNT</td>
<td>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</td>
<td>CNT[30:0]</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>
</tr>
<tr>
<td>0x28</td>
<td>TIMx_PSC</td>
<td>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</td>
<td>PSC[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>
</tr>
<tr>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td>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</td>
<td>ARR[31:0]</td>
<td>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 1 1 1</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>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</td>
<td>CCR1[31:0]</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>
</tr>
<tr>
<td>0x38</td>
<td>TIMx_CCR2</td>
<td>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</td>
<td>CCR2[31:0]</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>
</tr>
<tr>
<td>0x3C</td>
<td>TIMx_CCR3</td>
<td>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</td>
<td>CCR3[31:0]</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>
</tr>
<tr>
<td>0x40</td>
<td>TIMx_CCR4</td>
<td>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</td>
<td>CCR4[31:0]</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>
</tr>
<tr>
<td>0x44</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x48</td>
<td>TIMx_DCR</td>
<td>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</td>
<td>DBL[4:0]</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>
</tr>
<tr>
<td>0x4C</td>
<td>TIMx_DMAR</td>
<td>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</td>
<td>DMAB[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>
</tr>
<tr>
<td>0x50</td>
<td>TIM2_OR1</td>
<td>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</td>
<td>ETRSEL[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>
</tr>
<tr>
<td>0x60</td>
<td>TIM2_AF1</td>
<td>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</td>
<td>Reset value</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>
</tr>
</tbody>
</table>
Refer to Section 2.2 on page 66 for the register boundary addresses.
26 General-purpose timers (TIM16/TIM17)

26.1 TIM16/TIM17 introduction

The 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 TIM16/TIM17 timers are completely independent, and do not share any resources.

26.2 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
1. This signal can be used as trigger for some slave timer, see Section 26.3.18: Using timer output as trigger for other timers (TIM16/TIM17).
26.3 **TIM16/TIM17 functional description**

26.3.1 **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 254* and *Figure 255* give some examples of the counter behavior when the prescaler ratio is changed on the fly:
Figure 254. Counter timing diagram with prescaler division change from 1 to 2

Figure 255. Counter timing diagram with prescaler division change from 1 to 4
26.3.2 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). 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 256. Counter timing diagram, internal clock divided by 1

Figure 257. Counter timing diagram, internal clock divided by 2
**Figure 258. 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 259. 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 260. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

<table>
<thead>
<tr>
<th>CK_PSC</th>
<th>CEN</th>
<th>Timer clock = CK_CNT</th>
<th>Counter register</th>
<th>Counter overflow</th>
<th>Update event (UEV)</th>
<th>Update interrupt flag (UIF)</th>
<th>Auto-reload preload register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FF 36</td>
</tr>
</tbody>
</table>

Write a new value in TIMx_ARR

### Figure 261. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)

<table>
<thead>
<tr>
<th>CK_PSC</th>
<th>CEN</th>
<th>Timer clock = CK_CNT</th>
<th>Counter register</th>
<th>Counter overflow</th>
<th>Update event (UEV)</th>
<th>Update interrupt flag (UIF)</th>
<th>Auto-reload preload register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F5 36</td>
</tr>
</tbody>
</table>

Write a new value in TIMx_ARR
26.3.3 Repetition counter

Section 26.3.1: 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 262). 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.
26.3.4 Clock selection

The counter clock can be provided by the following clock sources:
- Internal clock (CK_INT)
- External clock mode1: external input pin

Internal 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 263 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

**Figure 263. Control circuit in normal mode, internal clock divided by 1**

![Diagram of 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 264. TI2 external clock connection example**

![Diagram of 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 265. Control circuit in external clock mode 1](MS31087V2)

### 26.3.5 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 266 to Figure 268* 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 266. 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 267. 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.

### 26.3.6 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 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 bit to 0 in the TIMx_CCRER 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_CCRER register.

7. If needed, enable the related interrupt request by setting the CC1E 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.
- CC1F 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 CC1E 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.

26.3.7 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.

26.3.8 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_CCMRx 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 269.
26.3.9 PWM mode

Pulse Width Modulation mode allows a signal to be generated 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 TIM16/TIM17 are capable of upcounting only. Refer to Upcounting mode on page 841.

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 270 shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.
26.3.10 Complementary outputs and dead-time insertion

The 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 169: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 876 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 271. Complementary output with dead-time insertion.**

**Figure 272. Dead-time waveforms with delay greater than the negative 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 26.4.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 879 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 a specific waveform to be sent (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.

**26.3.11 Using the break function**

The purpose of the break function is to protect power switches driven by PWM signals generated with the 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.

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.

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 GPIO (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 GPIO) 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 GPIO 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:** If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value.
If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

**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 the configuration of several parameters to be frozen (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 26.4.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 879. The LOCK bits can be written only once after an MCU reset.

The Figure 274 shows an example of behavior of the outputs in response to a break.
Figure 274. Output behavior in response to a break

- OCxREF
- OCx (OCxN not implemented, CCxP=0, OSx=1)
- OCx (OCxN not implemented, CCxP=0, OSx=0)
- OCx (OCxN not implemented, CCxP=1, OSx=1)
- OCx (OCxN not implemented, CCxP=1, OSx=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=1, 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, CCxNE=0, CCxNP=0, OISx=OISxN=0 or OISx=OISxN=1)
26.3.12 Bidirectional break inputs

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on Figure 275.

They allow the following:

- 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 BGINP and BKI 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 is active even if the BKDSRM bit is set and the open drain control is released. This prevents 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 168)

### Table 168. 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 275. Output redirection**

### 26.3.13 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 `tim_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 276* describes the behavior of the `tim_ocx` and `tim_ocxn` outputs when a COM event occurs, in 3 different examples of programmed configurations.
Figure 276. 6-step generation, COM example (OSSR=1)

Counter (CNT) (CCRx)

tim_ocxref

COM event

Example 1

Example 2

Example 3

CCxE = 1
CCxNE = 0
OCxM = 0010 (forced inactive)

Write OCxM to 0100
CCxE = 1
CCxNE = 0
OCxM = 0100

Write COM to 1

Write OCxM to 0101
CCxE = 0
CCxNE = 1
OCxM = 0101

Write CCxNE to 1 and OCxM to 0101
CCxE = 1
CCxNE = 0
OCxM = 0100 (forced inactive)

Write CCxNE to 0 and OCxM to 0100
CCxE = 1
CCxNE = 0
OCxM = 0010 (forced inactive)

Write CCxNE to 0
CCxE = 1
CCxNE = 0
OCxM = 0100

Write CCxNE to 1
CCxE = 1
CCxNE = 0
OCxM = 0010 (forced inactive)

Write CCxNE to 0
CCxE = 1
CCxNE = 0
OCxM = 0100

Write CCxNE to 1

MSv62343V1
26.3.14 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 277. 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 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.

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.

26.3.15 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 both the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. 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.

26.3.16 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.

26.3.17 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.

26.3.18 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

26.3.19 Debug mode

When the microcontroller enters debug mode (CPU1 Cortex®-M4 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 41.8.7: DBGMCU CPU1 APB2 peripheral freeze register (DBGMCU_APB2FZR).

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.
26.4 **TIM16/TIM17 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).

**26.4.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)**

Address offset: 0x00

Reset value: 0x0000

<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>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></td>
<td></td>
<td></td>
<td>r/w</td>
<td></td>
<td>r/w</td>
<td></td>
<td>r/w</td>
<td></td>
<td>r/w</td>
<td></td>
<td>r/w</td>
<td></td>
<td>r/w</td>
<td></td>
<td>r/w</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: \( t_{DTS} = t_{CK\_INT} \)
01: \( t_{DTS} = 2 \cdot t_{CK\_INT} \)
10: \( t_{DTS} = 4 \cdot t_{CK\_INT} \)
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.
26.4.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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_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).

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.
26.4.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

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 15:10 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
26.4.4  **TIMx status register (TIMx_SR)(x = 16 to 17)**

Address offset: 0x10
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:10</th>
<th>CC1OF: Capture/Compare 1 overcapture flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Must be kept at reset value.</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 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
  - 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 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.

### 26.4.5 TIM<sub>x</sub> event generation register (TIM<sub>x</sub>_EGR)(x = 16 to 17)

Address offset: 0x14  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</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>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></td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</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 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).
26.4.6 TIMx capture/compare mode register 1
(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:**

<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, 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=fDTS/2, N=
- 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 every 2 events
- 10: capture is done every 4 events
- 11: capture is done every 8 events
26.4.7 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:

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.
- **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*: 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).

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.

The OC1M[3] bit is not contiguous, located in bit 16.

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*: 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).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

- 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.

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).
26.4.8  TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

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></td>
<td></td>
<td></td>
<td></td>
<td></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 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**: 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).

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
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.

*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 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
0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

*When CC1 channel is configured as output*, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 169* for details.
### Table 169. 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>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>X</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</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 GPIO control and alternate function registers.

### 26.4.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24
26.4.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)
Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0 PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The counter clock frequency (CK_CNT) is equal to fCK_PSC / (PSC[15:0] + 1).</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>
</tr>
</tbody>
</table>

26.4.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)
Address offset: 0x2C
Reset value: 0xFFFF

<table>
<thead>
<tr>
<th>Bit 15:0 ARR[15:0]: Auto-reload value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARR is the value to be loaded in the actual auto-reload register.</td>
</tr>
<tr>
<td>Refer to the Section 26.3.1: Time-base unit on page 839 for more details about ARR update and behavior.</td>
</tr>
<tr>
<td>The counter is blocked while the auto-reload value is null.</td>
</tr>
</tbody>
</table>
26.4.12  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>
<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>

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 enabled, 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.

26.4.13  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>CCR1[15:0]</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>
<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 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).
26.4.14 TIMx break and dead-time register (TIMx_BDTR)\((x = 16 \text{ to } 17)\)

Address offset: 0x44
Reset value: 0x0000 0000

| Bit 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 | Reserved, must be kept at reset value. |
| 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 26.4.8: TIMx capture/compare enable register (TIMx_CCER)\((x = 16 \text{ to } 17)\) on page 874). |
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: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

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: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

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 26.4.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 874](#)).

- 0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO, 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 26.4.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 874).

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 CCxEn=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 tdtg with tdtg = tDTS
DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x tdtg with tdtg = 2 x tDTS
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x tdtg with tdtg = 8 x tDTS
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x tdtg with tdtg = 16 x tDTS

Example if tDTS = 125 ns (8 MHz), 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).

### 26.4.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

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>
<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>

STI
Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **DBL[4:0]**: DMA burst length
   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).
   
   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 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.
   
   Example:
   00000: TIMx_CR1,
   00001: TIMx_CR2,
   00010: TIMx_SMCR,
   ...

   **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.

### 26.4.16  TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

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 | rw | 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 (TIMx_CR1 address) + (DBA + DMA index) x 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).
### 26.4.17 TIM16 option register 1 (TIM16_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>
<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>T11_RMP[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 31:2: Reserved, must be kept at reset value.

Bits 1:0 **T11_RMP[1:0]:** Timer 16 input 1 connection
- This bit is set and cleared by software.
- 00: TIM16 TI1 is connected to GPIO
- 01: TIM16 TI1 is connected to LSI
- 10: TIM16 TI1 is connected to LSE
- 11: TIM16 TI1 is connected to RTC wake-up interrupt

### 26.4.18 TIM16 alternate function register 1 (TIM16_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>
<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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 31:12: Reserved, must be kept at reset value.

**Bit 11 BKCM P2P:** 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 BKCM P1P:** 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)*.

26.4.19 **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></td>
<td></td>
<td></td>
<td></td>
<td></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>rw</td>
<td>rw</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: TIM16_CH1 input
   Others: Reserved
26.4.20 TIM17 option register 1 (TIM17_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>

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

Bits 1:0 TI1_RMP[1:0]: Timer 17 input 1 connection
This bit is set and cleared by software.
00: TIM17 TI1 is connected to GPIO
01: TIM17 TI1 is connected to MSI
10: TIM17 TI1 is connected to HSE/32
11: TIM17 TI1 is connected to MCO

26.4.21 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 BKCM P2P: 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 BKCM P1P: 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).
26.4.22 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>
<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>TI1SEL[3:0]</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>
<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 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM17_CH1 input
Others: Reserved

Bit 9 **BKIN**: 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).*

Bits 31:4 Reserved, must be kept at reset value.
### 26.4.23 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

<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>
<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>0x00</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>TIMx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>TIMx_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Output Compare mode</td>
<td>OC1M[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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>TIMx_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>Input Capture mode</td>
<td>IC1F[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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>TIMx_CNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>CNT[15:0]</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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>TIMx_PSC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>PSC[15:0]</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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<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></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>TIMx_ARR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>ARR[15:0]</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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 170. TIM16/TIM17 register map and reset values
Table 170. 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   | 0 0 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   | TIMx_CCR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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 |
| 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 |
| 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 |
| 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 |
| 0x50   | TIM16_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 |
| 0x50   | TIM17_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 |
| 0x60   | TIM16_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 |
| 0x60   | TIM17_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 |
| 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 |
| 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 |

Refer to Section 2.2 on page 66 for the register boundary addresses.
27 Low-power timer (LPTIM)

27.1 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 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.

27.2 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: configurable internal clock source (see RCC section)
  – External clock source over LPTIM input (working with no embedded 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
27.3 LPTIM implementation

Table 171 and Table 172 describe LPTIM implementation: the full set of features is implemented in LPTIM1. LPTIM2 supports a smaller set of features, but is otherwise identical to LPTIM1.

Table 171. LPTIM features

<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>
<tr>
<td>External input clock</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wake-up from Stop</td>
<td>(2)</td>
<td>(3)</td>
</tr>
</tbody>
</table>

1. X = supported.
2. Wake-up supported from Stop 0, Stop 1 and Stop 2 modes.
3. Wake-up supported from Stop 0 and Stop 1 modes.

Table 172. LPTIM implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM1 external trigger</td>
<td>GPIO, RTC alarm A, RTC alarm B, RTC_TAMP[1:3] COMP1_OUT, COMP2_OUT</td>
<td>GPIO, RTC alarm A, RTC alarm B, RTC_TAMP2, COMP1_OUT, COMP2_OUT</td>
</tr>
<tr>
<td>LPTIM2 external trigger</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
27.4 LPTIM functional description

27.4.1 LPTIM block diagram

Figure 278. Low-power timer block diagram

1. lptim_in1 and lptim_in2 are respectively internal LPTIM input 1 and input 2 signals that can be connected to internal peripherals. lptim_out is the internal LPTIM output signal that can be connected to internal peripherals.

27.4.2 LPTIM trigger mapping

The LPTIM external trigger connections are detailed hereafter:

Table 173. 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</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>RTC_TAMP1 input detection</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>RTC_TAMP2 input detection</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>RTC_TAMP3 input detection</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>
27.4.3 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). 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 from configurable internal clock source (see RCC section).
- 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.

27.4.4 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), 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 279* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

**Figure 279. Glitch filter timing diagram**

![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.*

### 27.4.5 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 175. Prescaler 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>
<tr>
<td>011</td>
<td>/8</td>
</tr>
<tr>
<td>100</td>
<td>/16</td>
</tr>
<tr>
<td>101</td>
<td>/32</td>
</tr>
<tr>
<td>110</td>
<td>/64</td>
</tr>
<tr>
<td>111</td>
<td>/128</td>
</tr>
</tbody>
</table>
27.4.6 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.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

27.4.7 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 280.
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 281.

**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 282.

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.

### 27.4.8 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.

### 27.4.9 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 follows:

- 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 283 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 283. Waveform generation**

<table>
<thead>
<tr>
<th>LPTIM_ARR → Compare → 0 → PWM → One shot → Set once → PWM → One shot → Set once</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pol = 0</td>
</tr>
<tr>
<td>Pol = 1</td>
</tr>
</tbody>
</table>

### 27.4.10 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.

27.4.11 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 (PRES[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.

27.4.12 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.

### 27.4.13 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 pulses 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.

---

### 27.4.14 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 the counter. 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></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></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></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’).
27.4.15 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.

27.5 LPTIM low-power modes

Table 177. Effect of low-power modes on the LPTIM

<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>Stop</td>
<td>If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode (refer to Section 27.3: LPTIM implementation).</td>
</tr>
<tr>
<td>Standby</td>
<td>The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>
27.6 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 178. Interrupt events

<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</td>
<td>Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.</td>
</tr>
<tr>
<td>update OK</td>
<td></td>
</tr>
<tr>
<td>Compare register update</td>
<td>Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.</td>
</tr>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Direction change</td>
<td>Used in Encoder mode. Two interrupt flags are embedded to signal direction change:</td>
</tr>
<tr>
<td></td>
<td>– UP flag signals up-counting direction change</td>
</tr>
<tr>
<td></td>
<td>– DOWN flag signals down-counting direction change.</td>
</tr>
</tbody>
</table>

27.7 LPTIM registers

Refer to Section 1.2: List of abbreviations for registers for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).
### 27.7.1 LPTIM interrupt and status register (LPTIM_ISR)

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>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 31:7 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 27.3: 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 27.3: 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. CMPOK flag can be cleared by writing 1 to the CMPOKCF bit in the LPTIM_ICR register.

- **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. CMPM flag can be cleared by writing 1 to the CMPMCF bit in the LPTIM_ICR register.
### 27.7.2 LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:7</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DOWNCF: Direction change to down clear flag</td>
<td></td>
<td>Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.</td>
</tr>
<tr>
<td>5</td>
<td>UPCF: Direction change to UP clear flag</td>
<td></td>
<td>Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.</td>
</tr>
<tr>
<td>4</td>
<td>ARROKCF: Autoreload register update OK clear flag</td>
<td></td>
<td>Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register</td>
</tr>
<tr>
<td>3</td>
<td>CMPOKCF: Compare register update OK clear flag</td>
<td></td>
<td>Writing 1 to this bit clears the CMPOK flag in the LPTIM_ISR register</td>
</tr>
<tr>
<td>2</td>
<td>EXTTRIGCF: External trigger valid edge clear flag</td>
<td></td>
<td>Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register</td>
</tr>
<tr>
<td>1</td>
<td>ARRMCF: Autoreload match clear flag</td>
<td></td>
<td>Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register</td>
</tr>
<tr>
<td>0</td>
<td>CMPMCF: Compare match clear flag</td>
<td></td>
<td>Writing 1 to this bit clears the CMPM flag in the LPTIM_ISR register</td>
</tr>
</tbody>
</table>

### 27.7.3 LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:7</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>UPIE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ARROKIE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>CMPOKIE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>EXTTRIGIE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ARRMIE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CMPMIE:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:7 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 27.3: LPTIM implementation.*

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
- 0: UP interrupt disabled
- 1: UP interrupt enabled

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 27.3: LPTIM implementation.*

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
- 0: ARROK interrupt disabled
- 1: ARROK interrupt enabled

Bit 3 **CMPOKIE**: Compare register update OK Interrupt Enable
- 0: CMPOK interrupt disabled
- 1: CMPOK interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
- 0: EXTTRIG interrupt disabled
- 1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
- 0: ARRM interrupt disabled
- 1: ARRM interrupt enabled

Bit 0 **CMPMIE**: Compare match Interrupt Enable
- 0: CPM interrupt disabled
- 1: CPM interrupt enabled

**Caution:** The LPTIM_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’)

### 27.7.4 LPTIM configuration register (LPTIM_CFGR)

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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>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>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 31:30 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 27.3: 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_CNT and LPTIM_CMP registers  
1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CMP registers

Bit 20  **WAVE**: Waveform shape  
The WAVE bit controls the output shape  
0: Deactivate Set-once mode  
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 27.4.2: LPTIM 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').

27.7.5 LPTIM control register (LPTIM_CR)

Address offset: 0x010

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>ARE</td>
<td>COUN</td>
<td>TRST</td>
<td>CNT</td>
<td>STRT</td>
<td>SNG</td>
<td>STRT</td>
<td>ENA</td>
<td>BLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 4:3 CKFLT[1:0]: Configurable digital filter for external clock
The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.
00: any external clock signal level change is considered as a valid transition
01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 CKPOL[1:0]: Clock polarity
If LPTIM is clocked by an external clock source:
When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:
00: the rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.
01: the falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.
10: both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.
11: not allowed
Refer to Section 27.4.14: Encoder mode for more details about Encoder mode sub-modes.

Bit 0 CKSEL: Clock selector
The CKSEL bit selects which clock source the LPTIM will use:
0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)
1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CRGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').
Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE**: Reset after read enable
This bit is set and cleared by software. When RSTARE is set to ‘1’, any read access to LPTIM_CNT register will asynchronously reset LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

**Caution:** This bitfield is write-only. This means that the bit cannot be read back to verify the value which has been written. As an example, if this bit is set to 1, attempting to read it back will return 0 even if the "Reset after read" function is enabled (due to the fact that this bitfield has previously been written to 1). To turn off the "Reset after read" or to make sure that it has already been turned off, this bit should be reset (by programming it to 0) even if it already contains 0.

Bit 3 **COUNTRST**: Counter reset
This bit is set by software and cleared by hardware. When set to '1' this bit will trigger a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

**Caution:** COUNTRST must never be set to ‘1’ by software before it is already cleared to ‘0’ by hardware. Software should consequently check that COUNTRST bit is already cleared to ‘0’ before attempting to set it to ‘1’.

Bit 2 **CNTSTRT**: Timer start in Continuous mode
This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in Continuous mode.
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.
If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.
This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 1 **SNGSTRT**: LPTIM start in Single mode
This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in single pulse mode.
If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.
If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM_ARR and LPTIM_CNT registers.
This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 0 **ENABLE**: LPTIM enable
The ENABLE bit is set and cleared by software.
0:LPTIM is disabled
1:LPTIM is enabled
27.7.6 LPTIM compare register (LPTIM_CMP)

Address offset: 0x014  
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>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.

Bits 15:0 **CMP[15:0]:** Compare value

CMP is the compare value used by the LPTIM.

**Caution:** The LPTIM_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

27.7.7 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018  
Reset value: 0x0000 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>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.

Bits 15:0 **ARR[15:0]:** Auto reload value

ARR is the autoreload value for the LPTIM.  
This value must be strictly greater than the CMP[15:0] value.

**Caution:** The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).
27.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C
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>CNT[15:0]</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>
<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>

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

Bits 15:0 CNT[15:0]: Counter value

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.

27.7.9 LPTIM1 option register (LPTIM1_OR)

Address offset: 0x020
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>

<table>
<thead>
<tr>
<th>r</th>
<th>w</th>
</tr>
</thead>
</table>

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

Bit 1 OR_1: Option register bit 1
0: LPTIM1 input 2 is connected to I/O
1: LPTIM1 input 2 is connected to COMP2_OUT

Bit 0 OR_0: Option register bit 0
0: LPTIM1 input 1 is connected to I/O
1: LPTIM1 input 1 is connected to COMP1_OUT
27.7.10 LPTIM2 option register (LPTIM2_OR)

Address offset: 0x020
Reset value: 0x0000 0000

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

Bit 1 **OR_1**: Option register bit 1
- 0: LPTIM2 input 1 is connected to I/O
- 1: LPTIM2 input 1 is connected to COMP2_OUT

Bit 0 **OR_0**: Option register bit 0
- 0: LPTIM2 input 1 is connected to I/O
- 1: LPTIM2 input 1 is connected to COMP1_OUT
## 27.7.11 LPTIM register map

The following table summarizes the LPTIM registers.

### Table 179. LPTIM 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   | LPTIM_ISR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | LPTIM_ICR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | LPTIM_IER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | LPTIM_CFGR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | LPTIM_CR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | LPTIM_CMP     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x018  | LPTIM_ARR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | LPTIM_CNT     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | LPTIM_OR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

1. If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 27.3: LPTIM implementation.

Refer to Section 2.2 on page 66 for the register boundary addresses.
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 TIM16, and TIM17 as shown in Figure 285.

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.

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 generates the modulation envelope.

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_CFGR1 register and used to sink the high current needed to directly control an infrared LED.
29 Real-time clock (RTC)

29.1 Introduction

The RTC provides an automatic wake-up 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.

The RTC includes also a periodic programmable wake-up flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After Backup domain reset, all RTC registers are protected against possible parasitic write accesses.

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).
29.2 RTC main features

The RTC unit main features are the following (see Figure 286: RTC block diagram):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wake-up unit generating a periodic flag that triggers an automatic wake-up interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds.
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wake-up interrupt
  - Time-stamp
  - Tamper detection
- 20 backup registers.

29.3 RTC implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>STM32WB55xx</th>
<th>STM32WB35xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS pin</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>RTC_TAMP1</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>RTC_TAMP2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RTC_TAMP3</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>RTC_OUT on PC13</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>RTC_OUT on PB2</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
29.4 RTC functional description

29.4.1 RTC block diagram

The RTC includes:
- Tamper detection erases the backup registers.
- One timestamp event from I/O
- Tamper event detection can generate a timestamp event
- Timestamp can be generated when a switch to V\textsubscript{BAT} occurs
29.4.2 Clock and prescalers

The RTC clock source (RTCCLK) 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 8: 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 286: 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.

$f_{ck\_apre}$ is given by the following formula:

$$f_{ck\_apre} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

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.

$f_{ck\_spre}$ is given by the following formula:

$$f_{ck\_spre} = \frac{f_{RTCCLK}}{PREDIV\_S + 1 \cdot PREDIV\_A + 1}$$

The $ck\_spre$ clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see Section 29.4.5: Periodic auto-wake-up for details).

29.4.3 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 period, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see Section 29.7.4: RTC initialization and status).
register (RTC_ISR)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 1 RTCCLK period.

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 BYPSHAD 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 BYPSHAD=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.

29.4.4 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 RTC_ALARM output. RTC_ALARM output polarity can be configured through bit POL the RTC_CR register.

29.4.5 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input can be:
- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.
  - When RTCCLK is LSE(32.768 kHz), this allows to configure the wake-up interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spref (usually 1 Hz internal clock)
  - When ck_spref frequency is 1Hz, this allows to achieve a wake-up time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
    - from 1s to 18 hours when WUCKSEL [2:1] = 10
    - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 216 is added to the 16-bit counter current value. When the initialization sequence is complete (see Programming the wake-up timer on page 921), the timer starts counting down. When the wake-up 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_ISR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the RTC_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC.CR register. RTC_ALARM 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 wake-up timer.

29.4.6 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 clearing the DBP bit in the PWR_CR register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, all the RTC registers are write-protected. Writing to the 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 all the RTC registers except for RTC_TAMPCR, RTC_BKPxR, RTC.OR and RTC_ISR[31:8].

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_ISR 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_ISR 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_ISR 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 Backup 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_ISR 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 wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wake-up auto-reload value WUT[15:0], and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clock cycles after WUTE is cleared, due to clock synchronization.

### 29.4.7 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 APB 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 APB 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 APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCCLK cycle. 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 RTCCCLK period: 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 wake-up 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 920): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to Section 29.4.9: 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 RTCCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCCLK 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.

### 29.4.8 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ISR) 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 Backup 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 RTC tamper configuration register (RTC_TAMPCR), the RTC backup registers (RTC_BKPxR), the wake-up timer register (RTC_WUTR), the Alarm A and Alarm B registers (RTC_ALRMAASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR), and the Option register (RTC_OR).

In addition, when it is clocked by the LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to the RTC clock section of the Reset and clock controller for details on the list of RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

29.4.9 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.

29.4.10 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 synchronous 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
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

### 29.4.11 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 \((\text{cal\_cnt} = 0x20000/0x60000/0xA0000/0xE0000)\); and so on up to \(\text{CALM}[8]=1\) which causes 256 clocks to be masked \((\text{cal\_cnt} = 0xXX800)\).

While \(\text{CALM}\) allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit \(\text{CALP}\) can be used to increase the frequency by 488.5 ppm. Setting \(\text{CALP}\) to ‘1’ effectively inserts an extra RTCCCLK pulse every 2\(^{11}\) RTCCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using \(\text{CALM}\) together with \(\text{CALP}\), an offset ranging from -511 to +512 RTCCCLK 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 \((F_{\text{CAL}})\) given the input frequency \((F_{\text{RTCCLK}})\) is as follows:

\[
F_{\text{CAL}} = F_{\text{RTCCLK}} \times \left[1 + \left(\text{CALP} \times 512 - \text{CALM}\right) / \left(2^{20} + \text{CALM} - \text{CALP} \times 512\right)\right]
\]

**Calibration when PREDIV\_A<3**

The \(\text{CALP}\) bit can not be set to 1 when the asynchronous prescaler value (PREDIV\_A bits in RTC\_PRER register) is less than 3. If \(\text{CALP}\) was already set to 1 and PREDIV\_A bits are set to a value less than 3, \(\text{CALP}\) is ignored and the calibration operates as if \(\text{CALP}\) 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 RTCCCLK 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 RTCCCLK frequency of 32768 Hz, when \(\text{PREDIV\_A}\) equals 1 (division factor of 2), \(\text{PREDIV\_S}\) should be set to 16379 rather than 16383 (4 less). The only other interesting case is when \(\text{PREDIV\_A}\) equals 0, \(\text{PREDIV\_S}\) should be set to 32759 rather than 32767 (8 less).

If \(\text{PREDIV\_S}\) is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

\[
F_{\text{CAL}} = F_{\text{RTCCLK}} \times \left[1 + \left(256 - \text{CALM}\right) / \left(2^{20} + \text{CALM} - 256\right)\right]
\]

In this case, \(\text{CALM}[7:0]\) equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCCLK is exactly 32768.00 Hz.

**Verifying the RTC calibration**

RTC precision is ensured by measuring the precise frequency of RTCCCLK 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 RTCCCLK 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 RTCLK 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 RTCLK 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 RTCLK cycles over 8s). 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_ISR/INITF=0, by using the follow process:

1. Poll the RTC_ISR/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.

29.4.12 Time-stamp function

Time-stamp 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 time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a time-stamp event is detected on the RTC_TS pin.

When ITSE is set:

The calendar is saved in the time-stamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal time-stamp event is detected. The internal timestamp event is generated by the switch to the VBAT supply.

When a time-stamp event occurs, due to internal or external event, the time-stamp flag bit (TSF) in RTC_ISR register is set. In case the event is internal, the ITSF flag is also set in RTC_ISR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a time-stamp event occurs.

If a new time-stamp event is detected while the time-stamp flag (TSF) is already set, the time-stamp overflow flag (TSOVF) flag is set and the time-stamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.
**Note:** TSF is set 2 \( \text{ck\_apre} \) cycles after the time-stamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two time-stamp 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 time-stamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a time-stamp 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 time-stamp to be recorded. See the description of the TAMPTS control bit in [Section 29.7.16: RTC tamper configuration register (RTC_TAMPCR)](RM0434 Rev 13).

### 29.4.13 Tamper detection

The RTC_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for the following purposes:
- erase the RTC backup registers (default configuration)
- generate an interrupt, capable to wake-up from Stop and Standby modes
- generate a hardware trigger for the low-power timers

**RTC backup registers**

The backup registers (RTC_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 (see [Section 29.7.20: RTC backup registers (RTC_BKPxR)](RM0434 Rev 13) and [Tamper detection initialization on page 927]) except if the TAMPxNOERASE bit is set, or if TAMPxFM is set in the RTC_TAMPCR register.

**Tamper detection initialization**

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC_TAMPCR register.

Each RTC_TAMPx tamper detection input is associated with a flag TAMPxF in the RTC_ISR register.

When TAMPxFM is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:
- 3 \( \text{ck\_apre} \) cycles when TAMPFLLT differs from 0x0 (Level detection with filtering)
- 3 \( \text{ck\_apre} \) cycles when TAMPTS=1 (Timestamp on tamper event)
- No latency when TAMPFLLT=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 TAMPxFM is set:
A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 \text{ck}_{\text{rtc}} additional cycles.

By setting the TAMPIE bit in the RTC\_TAMPCR register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPIE is not allowed when one or more TAMPxMF is set.

When TAMPIE is cleared, each tamper pin event interrupt can be individually enabled by setting the corresponding TAMPxIE bit in the RTC\_TAMPCR register. Setting TAMPxIE is not allowed when the corresponding TAMPxMF is set.

**Trigger output generation on tamper event**

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMF bit is cleared in RTC\_TAMPCR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMF bit is set, the TAMPxF flag is masked, and kept cleared in RTC\_ISR 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.

**Timestamp on tamper event**

With TAMPTS set to ‘1’, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC\_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set at the same time that TSF or TSOVF is set.

**Edge detection on tamper inputs**

If the TAMPFLT bits are “00”, the RTC\_TAMPx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the RTC\_TAMPx 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 RTC\_TAMPx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC\_BKPxR). This prevents the application from writing to the backup registers while the RTC\_TAMPx input value still indicates a tamper detection. This is equivalent to a level detection on the RTC\_TAMPx input.

**Note:** Tamper detection is still active when V\text{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the RTC\_TAMPx is mapped should be externally tied to the correct level.
Level detection with filtering on RTC_TAMPx inputs

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 RTC_TAMPx 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 RTC_TAMPx 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 datasheets for the electrical characteristics of the pull-up resistors.

29.4.14 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The RTC_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 RTC_CALIB frequency is $f_{RTCCLK/(256 \times (PREDIV_A+1))}$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = 0x7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz. The 1 Hz output is affected when a shift operation is on going and may toggle during the shift operation (SHPF=1).

Note: When COSEL bit is cleared, the RTC_CALIB output is the output of the 6th stage of the asynchronous prescaler.

When COSEL bit is set, the RTC_CALIB output is the output of the 8th stage of the synchronous prescaler.

29.4.15 Alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output RTC_ALARM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Alarm output

The RTC_ALARM pin can be configured in output open drain or output push-pull using RTC_OR register.

Note: Once the RTC_ALARM output is enabled, it has priority over RTC_CALIB (COE bit is don’t care and must be kept cleared).
29.5 RTC low-power modes

Table 181. Effect of low-power modes on RTC

<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>Low-power run</td>
<td>No effect.</td>
</tr>
<tr>
<td>Low-power sleep</td>
<td>No effect. RTC interrupts cause the device to exit the Low-power sleep mode.</td>
</tr>
<tr>
<td>Stop 0</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Stop 1</td>
<td></td>
</tr>
<tr>
<td>Stop 2</td>
<td></td>
</tr>
<tr>
<td>Standby</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup 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 alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Shutdown mode.</td>
</tr>
</tbody>
</table>

29.6 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to Section 14: Extended interrupt and event controller (EXTI).

Table 182. Interrupt control bits

<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 mode</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm A</td>
<td>ALRAF</td>
<td>ALRAIE</td>
<td>Yes</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>Alarm B</td>
<td>ALRBF</td>
<td>ALRBIE</td>
<td>Yes</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>RTC_TS input (timestamp)</td>
<td>TSF</td>
<td>TSIE</td>
<td>Yes</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>RTC_TAMP1 input detection</td>
<td>TAMP1F</td>
<td>TAMPIE</td>
<td>Yes</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>RTC_TAMP2 input detection</td>
<td>TAMP2F</td>
<td>TAMPIE</td>
<td>Yes</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>RTC_TAMP3 input detection</td>
<td>TAMP3F</td>
<td>TAMPIE</td>
<td>Yes</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>Wakeup timer interrupt</td>
<td>WUTF</td>
<td>WUTIE</td>
<td>Yes</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
</tbody>
</table>

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.
29.7 RTC registers

Refer to Section 1.2 on page 61 of the reference manual for a list of abbreviations used in register descriptions.

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

29.7.1 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 920 and Reading the calendar on page 921.

This register is write protected. The write access procedure is described in RTC register write protection on page 920.

Address offset: 0x00
Backup domain reset value: 0x0000 0000
System reset: 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>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
29.7.2 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 920 and Reading the calendar on page 921.

This register is write protected. The write access procedure is described in RTC register write protection on page 920.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset: 0x0000 2101 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>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>
</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></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 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden

001: Monday

... 111: Sunday

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
29.7.3 RTC control register (RTC_CR)

Address offset: 0x08
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31:25</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 24</td>
<td>ITSE: timestamp on internal event enable</td>
</tr>
<tr>
<td></td>
<td>0: internal event timestamp disabled</td>
</tr>
<tr>
<td></td>
<td>1: internal event timestamp enabled</td>
</tr>
<tr>
<td>Bit 23</td>
<td>COE: Calibration output enable</td>
</tr>
<tr>
<td></td>
<td>0: Calibration output disabled</td>
</tr>
<tr>
<td></td>
<td>1: Calibration output enabled</td>
</tr>
<tr>
<td>Bits 22:21</td>
<td>OSEL[1:0]: Output selection</td>
</tr>
<tr>
<td></td>
<td>These bits are used to select the flag to be routed to RTC_ALARM output</td>
</tr>
<tr>
<td></td>
<td>00: Output disabled</td>
</tr>
<tr>
<td></td>
<td>01: Alarm A output enabled</td>
</tr>
<tr>
<td></td>
<td>10: Alarm B output enabled</td>
</tr>
<tr>
<td></td>
<td>11: Wake-up output enabled</td>
</tr>
<tr>
<td>Bit 20</td>
<td>POL: Output polarity</td>
</tr>
<tr>
<td></td>
<td>This bit is used to configure the polarity of RTC_ALARM output</td>
</tr>
<tr>
<td></td>
<td>0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])</td>
</tr>
<tr>
<td></td>
<td>1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).</td>
</tr>
<tr>
<td>Bit 19</td>
<td>COSEL: Calibration output selection</td>
</tr>
<tr>
<td></td>
<td>When COE=1, this bit selects which signal is output on RTC_CALIB.</td>
</tr>
<tr>
<td></td>
<td>0: Calibration output is 512 Hz (with default prescaler setting)</td>
</tr>
<tr>
<td></td>
<td>1: Calibration output is 1 Hz (with default prescaler setting)</td>
</tr>
<tr>
<td></td>
<td>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 29.4.14: Calibration clock output</td>
</tr>
<tr>
<td>Bit 18</td>
<td>BKP: Backup</td>
</tr>
<tr>
<td></td>
<td>This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.</td>
</tr>
</tbody>
</table>
Bit 17 **SUB1H**: Subtract 1 hour (winter time change)
When this bit is set, 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 outside initialization mode.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set, 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 outside initialization mode.

Bit 15 **TSIE**: Time-stamp interrupt enable
0: Time-stamp Interrupt disable
1: Time-stamp Interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable
0: Wake-up timer interrupt disabled
1: Wake-up 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**: Wake-up timer enable
0: Wake-up timer disabled
1: Wake-up timer enabled

*Note: When the wake-up 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.

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 APB clock is less than seven times the frequency of RTCCLK, BYPHAD must be set to ‘1’.

Bit 4  **REFCKON**: 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**: Time-stamp event active edge
0: RTC_TS input rising edge generates a time-stamp event
1: RTC_TS input falling edge generates a time-stamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0  **WUCKSEL[2:0]**: Wake-up 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 (see note below)

*Note:* Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).

WUT = Wake-up 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_ISR WUTF 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.

*This register is write protected. The write access procedure is described in RTC register write protection on page 920."

**Caution:** TSE must be reset when TSEDGE is changed to avoid spuriously setting of TSF.
29.7.4 RTC initialization and status register (RTC_ISR)

Address offset: 0x0C
Backup domain reset value: 0x0000 0007
System reset: not affected except INIT, INITF, and RSF bits which are cleared to ‘0’

<table>
<thead>
<tr>
<th>31 30</th>
<th>29 28</th>
<th>27 26</th>
<th>25 24</th>
<th>23 22</th>
<th>21 20</th>
<th>19 18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w0</td>
<td>r</td>
<td>rc_w0</td>
<td>r</td>
<td>rc_w0</td>
<td>r</td>
<td>rc_w0</td>
<td>r</td>
<td>rc_w0</td>
</tr>
<tr>
<td>15 14</td>
<td>13 12</td>
<td>11 10</td>
<td>9 8</td>
<td>7 6</td>
<td>5 4</td>
<td>3 2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TAMP3F</td>
<td>TAMP2F</td>
<td>TAMP1F</td>
<td>TSOVF</td>
<td>TSF</td>
<td>WUTF</td>
<td>ALRBF</td>
<td>ALRAF</td>
<td>INIT</td>
</tr>
</tbody>
</table>

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 ITSF: Internal time-stamp flag
This flag is set by hardware when a time-stamp on the internal event occurs.
This flag is cleared by software by writing 0, and must be cleared together with TSF bit by writing 0 in both bits.

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.

Bit 15 TAMP3F: RTC_TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP3 input.
This is cleared by software writing 0.

Bit 14 TAMP2F: RTC_TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP2 input.
This is cleared by software writing 0.

Bit 13 TAMP1F: RTC_TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the RTC_TAMP1 input.
This is cleared by software writing 0.

Bit 12 TSOVF: Time-stamp overflow flag
This flag is set by hardware when a time-stamp event occurs while TSF is already set.
This is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 11 TSF: Time-stamp flag
This flag is set by hardware when a time-stamp event occurs.
This is cleared by software by writing 0. If ITSF flag is set, TSF must be cleared together with ITSF by writing 0 in both bits.
Bit 10 **WUTF**: Wake-up timer flag  
This flag is set by hardware when the wake-up auto-reload counter reaches 0.  
This flag is cleared by software by writing 0.  
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 9 **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).  
This flag is cleared by software by writing 0.

Bit 8 **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).  
This flag is cleared by software by writing 0.

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.  
*Note: This bit is write protected. The write access procedure is described in RTC register write protection on page 920.*

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_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSSHAD=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  
*Note: This bit is write protected. The write access procedure is described in RTC register write protection on page 920.*

Bit 4 **INITS**: Initialization status flag  
This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).  
0: Calendar has not been initialized  
1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending  
0: No shift operation is pending  
1: A shift operation is 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.
Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware up to 2 RTCCLK cycles after the WUTE bit has been set to 0 in RTC_CR, and is cleared up to 2 RTCCLK cycles after the WUTE bit has been set to 1. The wake-up timer values can be changed when WUTE bit is cleared and WUTWF is set.

0: Wake-up timer configuration update not allowed
1: Wake-up 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

**Note:** The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.
29.7.5 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 920.

This register is write protected. The write access procedure is described in RTC register write protection on page 920.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

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></td>
<td></td>
<td></td>
<td></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>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>Prediv_S[14:0]</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>
</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:
ck_apre frequency = RTCCLK 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:
ck_spre frequency = ck_apre frequency/(Prediv_S+1)
29.7.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ISR.

This register is write protected. The write access procedure is described in RTC register write protection on page 920.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0 | WUT[15:0]: Wake-up auto-reload value bits |

When the wake-up 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 wake-up 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 (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.
29.7.7  **RTC alarm A register (RTC_ALRMAR)**

This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 920](#).

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Backup Domain Reset Value</th>
<th>System Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1C</td>
<td>0x0000 0000</td>
<td>not affected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>MSK4: Alarm A date mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm A set if the date/day match</td>
</tr>
<tr>
<td>1</td>
<td>Date/day don’t care in Alarm A comparison</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>WDSEL: Week day selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DU[3:0] represents the date units</td>
</tr>
<tr>
<td>1</td>
<td>DU[3:0] represents the week day. DT[1:0] is don’t care.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 29:28</th>
<th>DT[1:0]: Date tens in BCD format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 27:24</td>
<td>DU[3:0]: Date units or day in BCD format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>MSK3: Alarm A hours mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm A set if the hours match</td>
</tr>
<tr>
<td>1</td>
<td>Hours don’t care in Alarm A comparison</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 22</th>
<th>PM: AM/PM notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AM or 24-hour format</td>
</tr>
<tr>
<td>1</td>
<td>PM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 21:20</th>
<th>HT[1:0]: Hour tens in BCD format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 19:16</td>
<td>HU[3:0]: Hour units in BCD format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>MSK2: Alarm A minutes mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm A set if the minutes match</td>
</tr>
<tr>
<td>1</td>
<td>Minutes don’t care in Alarm A comparison</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 14:12</th>
<th>MNT[2:0]: Minute tens in BCD format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 11:8</td>
<td>MNU[3:0]: Minute units in BCD format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>MSK1: Alarm A seconds mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm A set if the seconds match</td>
</tr>
<tr>
<td>1</td>
<td>Seconds don’t care in Alarm A comparison</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 6:4</th>
<th>ST[2:0]: Second tens in BCD format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 3:0</td>
<td>SU[3:0]: Second units in BCD format.</td>
</tr>
</tbody>
</table>
29.7.8 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 920.

Address offset: 0x20
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>MSK4: Alarm B date mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm B set if the date and day match</td>
</tr>
<tr>
<td>1</td>
<td>Date and day don’t care in Alarm B comparison</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>WDSEL: Week day selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DU[3:0] represents the date units</td>
</tr>
<tr>
<td>1</td>
<td>DU[3:0] represents the week day. DT[1:0] is don’t care.</td>
</tr>
</tbody>
</table>

Bits 29:28 DT[1:0]: Date tens in BCD format

Bits 27:24 DU[3:0]: Date units or day in BCD format

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>MSK3: Alarm B hours mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm B set if the hours match</td>
</tr>
<tr>
<td>1</td>
<td>Hours don’t care in Alarm B comparison</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 22</th>
<th>PM: AM/PM notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AM or 24-hour format</td>
</tr>
<tr>
<td>1</td>
<td>PM</td>
</tr>
</tbody>
</table>

Bits 21:20 HT[1:0]: Hour tens in BCD format

Bits 19:16 HU[3:0]: Hour units in BCD format

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>MSK2: Alarm B minutes mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm B set if the minutes match</td>
</tr>
<tr>
<td>1</td>
<td>Minutes don’t care in Alarm B comparison</td>
</tr>
</tbody>
</table>

Bits 14:12 MNT[2:0]: Minute tens in BCD format

Bits 11:8 MNU[3:0]: Minute units in BCD format

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>MSK1: Alarm B seconds mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm B set if the seconds match</td>
</tr>
<tr>
<td>1</td>
<td>Seconds don’t care in Alarm B comparison</td>
</tr>
</tbody>
</table>

Bits 6:4 ST[2:0]: Second tens in BCD format

Bits 3:0 SU[3:0]: Second units in BCD format
### 29.7.9 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>
</tbody>
</table>

15  | 14  | 13  | 12  | 11  | 10  |  9  |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |

|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

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.

### 29.7.10 RTC sub second register (RTC_SSR)

Address offset: 0x28  
Backup domain reset value: 0x0000 0000  
System reset: 0x0000 0000 when BYPASHAD = 0. Not affected when BYPASHAD = 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>

15  | 14  | 13  | 12  | 11  | 10  |  9  |  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
| r   | r   | r   | r   | r   | r   | r   | r   | r   | r   | r   | r   | r   |     |     |     |

Bits 31:16 Reserved, must be kept at reset value.  
Bits 15:0 **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.
29.7.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in RTC register write protection on page 920.

Address offset: 0x2C

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>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>
</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>Res</td>
<td>SUBFS[14:0]</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>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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **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_ISR).

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.

Bits 14:0 **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_ISR).

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:

\[
\text{Delay (seconds)} = \frac{\text{SUBFS}}{\text{PREDIV_S} + 1}
\]

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:

\[
\text{Advance (seconds)} = (1 - \frac{\text{SUBFS}}{\text{PREDIV_S} + 1})
\]

**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.
### 29.7.12 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

**Address offset**: 0x30

**Backup domain reset value**: 0x0000 0000

**System reset**: not affected

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:23</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>PM</td>
<td>AM/PM notation</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>Reserved</td>
<td>Must be kept at reset value.</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>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>3:0</td>
<td>SU[3:0]</td>
<td>Second units in BCD format.</td>
</tr>
</tbody>
</table>
### 29.7.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15:13</td>
<td>WDU[2:0]: Week day units</td>
</tr>
<tr>
<td></td>
<td>Bit 12 MT: Month tens in BCD format</td>
</tr>
<tr>
<td>11:8</td>
<td>MU[3:0]: Month units in BCD format</td>
</tr>
<tr>
<td></td>
<td>Bits 7:6 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5:4</td>
<td>DT[1:0]: Date tens in BCD format</td>
</tr>
<tr>
<td>3:0</td>
<td>DU[3:0]: Date units in BCD format</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></td>
<td></td>
<td></td>
<td></td>
<td></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></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>
</tr>
</tbody>
</table>

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
29.7.14 RTC time-stamp sub second register (RTC_TSSSR)

The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

Address offset: 0x38
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>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>

SS[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.
29.7.15 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in RTC register write protection on page 920.

Address offset: 0x3C
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>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 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 29.4.11: 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 29.4.11: 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 29.4.11: 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 \(2^{20}\) RTCCCLK 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 29.4.11: RTC smooth digital calibration on page 924.
29.7.16  RTC tamper configuration register (RTC_TAMPCR)

Address offset: 0x40
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>TAMP3 MF</th>
<th>TAMP3 NO ERASE</th>
<th>TAMP3 IE</th>
<th>TAMP2 MF</th>
<th>TAMP2 NO ERASE</th>
<th>TAMP2 IE</th>
<th>TAMP1 MF</th>
<th>TAMP1 NO ERASE</th>
<th>TAMP1 IE</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
</tbody>
</table>

Bits 31:25  Reserved, must be kept at reset value.

Bit 24  **TAMP3MF**: Tamper 3 mask flag
0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers are not erased.

*Note: The Tamper 3 interrupt must not be enabled when TAMP3MF is set.*

Bit 23  **TAMP3NOERASE**: Tamper 3 no erase
0: Tamper 3 event erases the backup registers.
1: Tamper 3 event does not erase the backup registers.

Bit 22  **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt is disabled if TAMPIE = 0.
1: Tamper 3 interrupt enabled.

Bit 21  **TAMP2MF**: Tamper 2 mask flag
0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers are not erased.

*Note: The Tamper 2 interrupt must not be enabled when TAMP2MF is set.*

Bit 20  **TAMP2NOERASE**: Tamper 2 no erase
0: Tamper 2 event erases the backup registers.
1: Tamper 2 event does not erase the backup registers.

Bit 19  **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt is disabled if TAMPIE = 0.
1: Tamper 2 interrupt enabled.

Bit 18  **TAMP1MF**: Tamper 1 mask flag
0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers are not erased.

*Note: The Tamper 1 interrupt must not be enabled when TAMP1MF is set.*
Bit 17 **TAMP1NOERASE**: Tamper 1 no erase
0: Tamper 1 event erases the backup registers.
1: Tamper 1 event does not erase the backup registers.

Bit 16 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt is disabled if TAMPIE = 0.
1: Tamper 1 interrupt enabled.

Bit 15 **Tamppudis**: RTC_TAMPx pull-up disable
This bit determines if each of the RTC_TAMPx pins are precharged before each sample.
0: Precharge RTC_TAMPx pins before sampling (enable internal pull-up)
1: Disable precharge of RTC_TAMPx pins.

Bits 14:13 **Tampprch[1:0]**: RTC_TAMPx precharge duration
These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the RTC_TAMPx inputs.
0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 12:11 **Tampflt[1:0]**: RTC_TAMPx filter count
These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a Tamper event. TAMPFLT is valid for each of the RTC_TAMPx inputs.
0x0: Tamper event is activated on edge of RTC_TAMPx input transitions to the active level (no internal pull-up on RTC_TAMPx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 10:8 **Tampfreq[2:0]**: Tamper sampling frequency
Determines the frequency at which each of the RTC_TAMPx inputs are sampled.
0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Bit 7 **Tampts**: Activate timestamp on tamper detection event
0: Tamper detection event does not cause a timestamp to be saved
1: Save timestamp on tamper detection event
TAMPTS is valid even if TSE=0 in the RTC_CR register.

Bit 6 **Tamp3trg**: Active level for RTC_TAMP3 input
if TAMPFLT ≠ 00:
0: RTC_TAMP3 input staying low triggers a tamper detection event.
1: RTC_TAMP3 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP3 input rising edge triggers a tamper detection event.
1: RTC_TAMP3 input falling edge triggers a tamper detection event.
Caution: When TAMPFLT = 0, TAMPxE must be reset when TAMPxTRG is changed to avoid spuriously setting TAMPxF.

Bit 5 TAMP3E: RTC_TAMP3 detection enable
0: RTC_TAMP3 input detection disabled
1: RTC_TAMP3 input detection enabled

Bit 4 TAMP2TRG: Active level for RTC_TAMP2 input
if TAMPFLT != 00:
0: RTC_TAMP2 input staying low triggers a tamper detection event.
1: RTC_TAMP2 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP2 input rising edge triggers a tamper detection event.
1: RTC_TAMP2 input falling edge triggers a tamper detection event.

Bit 3 TAMP2E: RTC_TAMP2 input detection enable
0: RTC_TAMP2 detection disabled
1: RTC_TAMP2 detection enabled

Bit 2 TAMPIE: Tamper interrupt enable
0: Tamper interrupt disabled
1: Tamper interrupt enabled.

Note: This bit enables the interrupt for all tamper pins events, whatever TAMPxIE level. If this bit is cleared, each tamper event interrupt can be individually enabled by setting TAMPxIE.

Bit 1 TAMP1TRG: Active level for RTC_TAMP1 input
If TAMPFLT != 00
0: RTC_TAMP1 input staying low triggers a tamper detection event.
1: RTC_TAMP1 input staying high triggers a tamper detection event.
If TAMPFLT = 00:
0: RTC_TAMP1 input rising edge triggers a tamper detection event.
1: RTC_TAMP1 input falling edge triggers a tamper detection event.

Bit 0 TAMP1E: RTC_TAMP1 input detection enable
0: RTC_TAMP1 detection disabled
1: RTC_TAMP1 detection enabled
29.7.17 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 920

Address offset: 0x44

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>
</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.

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.
29.7.18 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: 0x48
Backup domain reset value: 0x0000 0000
System reset: not affected

| 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. |
| ... |
| 0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared. |
| 0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared. |
| 0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared. |
| 0xF: 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. |
29.7.19 RTC option register (RTC_OR)

Address offset: 0x4C
Backup domain reset value: 0x0000 0000
System reset: not affected

| Bit 31:2 | Reserved, must be kept at reset value. |
| Bit 1   | **RTC_OUT_RMP**: RTC_OUT remap          |
| Setting this bit allows to remap the RTC outputs on PB2 as follows: |
| **RTC_OUT_RMP = '0' (only available on STM32WB55):** |
| If OSEL /= '00': RTC_ALARM is output on PC13 |
| If OSEL = '00' and COE = '1': RTC_CALIB is output on PC13 |
| **RTC_OUT_RMP = '1':** |
| If OSEL /= '00' and COE = '0': RTC_ALARM is output on PB2 |
| If OSEL = '00' and COE = '1': RTC_CALIB is output on PB2 |
| If OSEL /= '00' and COE = '1': RTC_CALIB is output on PB2 and (on STM32WB55 only) RTC_ALARM is output on PC13. |

| Bit 0   | **RTC_ALARM_TYPE**: RTC_ALARM output type on PC13 |
| This bit is set and cleared by software |
| 0: RTC_ALARM, when mapped on PC13, is open-drain output |
| 1: RTC_ALARM, when mapped on PC13, is push-pull output |

Note: This bit is reserved for STM32WB35.

29.7.20 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x9C
Backup domain reset value: 0x0000 0000
System reset: not affected
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. This register is reset on a tamper detection event, as long as TAMPxF=1.

### 29.7.21 RTC register map

#### Table 183. RTC 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   | RTC_TR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x04   | RTC_DR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | RTC_CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x0C   | RTC_ISR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x10   | RTC_PRER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 1  | 1  | 1  | 1  | 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  |
| 0x14   | RTC_WUTR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  | 1  |
| 0x1C   | 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  |
| 0x20   | RTC_ALRMBR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x24   | RTC_WPR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0x28   | RTC_SSR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |

---

ST

RM0434 Rev 13

955/1530
Refer to Section 2.2 on page 66 for the register boundary addresses.
30 Independent watchdog (IWDG)

30.1 Introduction

The devices feature an embedded watchdog peripheral (IWDG) that offers a combination of high safety level, timing accuracy, and flexibility of use. This peripheral detects and solves malfunctions due to software failure, and triggers a system reset when the counter reaches a given timeout value.

The independent watchdog is clocked by its own dedicated low-speed clock (LSI), and 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 31: System window watchdog (WWDG).

30.2 IWDG main features

- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog is activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog is activated) if the downcounter is reloaded outside the window

30.3 IWDG functional description

30.3.1 IWDG block diagram

Figure 287 shows the functional blocks of the independent watchdog module.

Figure 287. Independent watchdog block diagram

1. The register interface is located in the V_{CORE} voltage domain. The watchdog function is located in the V_{DD} 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.

### 30.3.2 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)**, 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 to reset the downcounter to the **IWDG reload register (IWDG_RLR)** value, and to 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 the counter value to be refreshed by the RLR when the **IWDG status register (IWDG_SR)** is set to 0x0000 0000.

#### Configuring the IWDG when the window option is disabled

When the window option 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).
### 30.3.3 Hardware watchdog

If this 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 the end of count, or if the downcounter is reloaded inside the window.

### 30.3.4 Low-power freeze

Depending on the IWDG_STOP and IWDG_STBY options configuration, the IWDG can continue counting or not during the Stop mode and the Standby mode, respectively. If the IWDG is kept running during Stop or Standby modes, it can wake up the device from this mode. Refer to *User and read protection option bytes* for more details.

### 30.3.5 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 it, first write the code 0x0000 5555 in the **IWDG key register (IWDG_KR)**. A write access to this register with a different value breaks the sequence, and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an ongoing update of the prescaler, or of the downcounter reload value, or of the window value.

### 30.3.6 Debug mode

When the CPU1 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.
30.4 IWDG registers

Refer to Section 1.2 on page 61 for a list of abbreviations used in register descriptions.

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

30.4.1 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>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]**

w w w w w w w w w w w w w w w w

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 30.3.5: Register access protection)

Writing the key value 0xC000 starts the watchdog (except if the hardware watchdog option is selected)
30.4.2 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 30.3.5: 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 VDD 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.
30.4.3  **IWDG reload register (IWDG_RLR)**

Address offset: 0x08  
Reset value: 0x0000 0FFF (reset by Standby 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>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.

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 VDD 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.*
### 30.4.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not 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>
</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 31:3 Reserved, must be kept at reset value.

**Bit 2 WVU:** 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\textsubscript{DD} voltage domain (takes up to five prescaled clock cycles).

Window value can be updated only when WVU bit is reset.

**Bit 1 RVU:** 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\textsubscript{DD} voltage domain (takes up to five prescaled clock cycles).

Reload value can be updated only when RVU bit is reset.

**Bit 0 PVU:** 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\textsubscript{DD} voltage domain (takes up to five LSI clock 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.
### 30.4.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby 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></td>
<td></td>
<td></td>
<td></td>
<td></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>WIN[11:0]</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>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected, see Section 30.3.5, 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.
30.4.6  IWDG register map

The following table gives the IWDG 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>IWDG_KR</td>
<td>0x04</td>
<td>IWDG_PR</td>
</tr>
<tr>
<td>0x08</td>
<td>IWDG_RLR</td>
<td>0x0C</td>
<td>IWDG_SR</td>
</tr>
<tr>
<td>0x10</td>
<td>IWDG_WINR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.2 on page 66 for the register boundary addresses.
31 System window watchdog (WWDG)

31.1 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 down-counter before the T6 bit is cleared. An MCU reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches 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 requiring the watchdog to react within an accurate timing window.

31.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see Figure 289)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

31.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (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 can take place only when the counter value is lower than or equal to 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 288 for the WWDG block diagram.
31.3.1 WWDG block diagram

Figure 288. Watchdog block diagram

31.3.2 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.

31.3.3 Controlling the down-counter

This down-counter 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 that represent 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 289). The WWDG configuration register (WWDG_CFR) contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. Figure 289 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).

31.3.4 How to program the watchdog timeout

Use the formula in Figure 289 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.

The formula to calculate the timeout value is given by:

\[ t_{\text{WWDG}} = t_{\text{PCLK}} \times \frac{4096}{2^{\text{WDGTB}[2:0]}} \times T_{5:0} + 1 \text{ ms} \]

where:

- \( t_{\text{WWDG}} \): WWDG timeout
- \( t_{\text{PCLK}} \): APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

\[ t_{\text{WWDG}} = 1 \times 48000 \times \frac{4096}{2^3} \times 63 + 1 = 43.69 \text{ ms} \]

Refer to the datasheet for the minimum and maximum values of \( t_{\text{WWDG}} \).
31.3.5 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 41: Debug support (DBG).

31.4 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog 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 ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

*Note:* When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

31.5 WWDG registers

Refer to Section 1.2 on page 61 for a list of abbreviations used in register descriptions.

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

31.5.1 WWDG 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>
<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>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.
31.5.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004
Reset value: 0x0000 007F

<table>
<thead>
<tr>
<th>Bit 31:14</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 13:11</td>
<td><strong>WDGTB[2:0]:</strong> Timer base</td>
</tr>
<tr>
<td>The timebase of the prescaler can be modified as follows:</td>
<td></td>
</tr>
<tr>
<td>000: CK counter clock (PCLK div 4096) div 1</td>
<td></td>
</tr>
<tr>
<td>001: CK counter clock (PCLK div 4096) div 2</td>
<td></td>
</tr>
<tr>
<td>010: CK counter clock (PCLK div 4096) div 4</td>
<td></td>
</tr>
<tr>
<td>011: CK counter clock (PCLK div 4096) div 8</td>
<td></td>
</tr>
<tr>
<td>100: CK counter clock (PCLK div 4096) div 16</td>
<td></td>
</tr>
<tr>
<td>101: CK counter clock (PCLK div 4096) div 32</td>
<td></td>
</tr>
<tr>
<td>110: CK counter clock (PCLK div 4096) div 64</td>
<td></td>
</tr>
<tr>
<td>111: CK counter clock (PCLK div 4096) div 128</td>
<td></td>
</tr>
<tr>
<td>Bit 10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9</td>
<td><strong>EWI:</strong> Early wake-up interrupt enable</td>
</tr>
<tr>
<td>Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.</td>
<td></td>
</tr>
<tr>
<td>Bits 8:7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 6:0</td>
<td><strong>W[6:0]:</strong> 7-bit window value</td>
</tr>
<tr>
<td>These bits contain the window value to be compared with the down-counter.</td>
<td></td>
</tr>
</tbody>
</table>
31.5.3  **WWDG status register (WWDG_SR)**

Address offset: 0x008  
Reset value: 0x0000 0000

<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>
<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>0x000</td>
<td>WWDG_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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0x004</td>
<td>WWDG_CFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0x008</td>
<td>WWDG_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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** **EWIF**: Early wake-up 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.

31.5.4  **WWDG register map**

The following table gives the WWDG register map and 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>
<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>0x000</td>
<td>WWDG_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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0x004</td>
<td>WWDG_CFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0x008</td>
<td>WWDG_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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 on page 66** for the register boundary addresses.
32 Inter-integrated circuit (I2C) interface

32.1 Introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) 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+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

32.2 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
- Clock stretching (optional)

1-byte buffer with DMA capability

Programmable analog and digital noise filters

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

Wake-up from Stop mode on address match

For information on I2C instantiation, refer to Section 32.3: I2C implementation.
32.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 186. I2C implementation

<table>
<thead>
<tr>
<th>I2C features(1)</th>
<th>I2C1</th>
<th>I2C3</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 20 mA 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>X</td>
</tr>
<tr>
<td>Wake-up from Stop 0 / Stop 1 mode on address match</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wake-up from Stop 2 mode on address match</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>SMBus/PMBus</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X = supported.

32.4 I2C functional description

In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the PCLK1 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).
32.4.1 I2C block diagram

Figure 290. Block diagram

32.4.2 I2C pins and internal signals

Table 187. I2C input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_SDA</td>
<td>Bidirectional</td>
<td>I²C-bus data</td>
</tr>
<tr>
<td>I2C_SCL</td>
<td>Bidirectional</td>
<td>I²C-bus clock</td>
</tr>
<tr>
<td>I2C_SMBA</td>
<td>Bidirectional</td>
<td>SMBus alert</td>
</tr>
</tbody>
</table>
32.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period $t_{I2CCLK}$ must respect the following conditions:

$$t_{I2CCLK} < \left( t_{LOW} - t_{filters} \right) / 4$$
$$t_{I2CCLK} < t_{HIGH}$$

where $t_{LOW}$ is the SCL low time, $t_{HIGH}$ is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is $DNF[3:0] \times t_{I2CCLK}$.

The PCLK1 clock period $t_{PCLK}$ must respect the condition $t_{PCLK} < 4/3 \times t_{SCL}$, where $t_{SCL}$ is the SCL period.

Caution: When the I2C kernel is clocked by PCLK1, this clock must respect the conditions for $t_{I2CCLK}$.

32.4.4 Mode selection

The peripheral can operate as:
- slave transmitter
- slave receiver
- master transmitter
- master receiver

By default, the peripheral operates in slave mode. It automatically switches from slave to master mode upon generating START condition, and from master to slave mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multimaster I²C-bus environment.

Communication flow

In master mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In slave mode, the peripheral recognizes its own 7-bit or 10-bit address, 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 one (7-bit addressing) or two (10-bit addressing) bytes following the START condition contain the address. The address is always transmitted in master mode.
The following figure shows the transmission of a single byte. The master generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

![I2C-bus protocol](image)

The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

### 32.4.5 I2C initialization

#### Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the clock controller, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register. Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to Section 32.4.6 for more details.

#### Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of I2CCLK. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen I2CCLK periods.

The following table compares the two filters.
### Caution:
The filter configuration cannot be changed when the I2C peripheral is enabled.

### I2C Timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.

---

#### Table 189. Comparison of analog and digital filters

<table>
<thead>
<tr>
<th>Item</th>
<th>Analog filter</th>
<th>Digital filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering capacity&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>≥ 50 ns</td>
<td>One to fifteen I2CCLK periods</td>
</tr>
<tr>
<td>Benefits</td>
<td>Available in Stop mode</td>
<td>– Programmable filtering capacity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Extra filtering capability versus I²C-bus specification requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Stable filtering capacity</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>Filtering capacity variation with temperature, voltage, and silicon process</td>
<td>Wake-up from Stop mode on address match not supported when the digital filter is enabled</td>
</tr>
</tbody>
</table>

1. Maximum duration of spikes that the filter can suppress
When the SCL falling edge is internally detected, the delay $t_{SDADEL}$ (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

$$t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK},$$

where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$.

The total SDA output delay is:

$$t_{SYNC1} + ([SDADEL \times (PRESC + 1) + 1] \times t_{I2CCLK}).$$

The $t_{SYNC1}$ duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay $t_{DNF} = DNF \times t_{I2CCLK}$ introduced by the digital filter (if enabled)
- delay due to SCL synchronization to I2CCLK clock (two to three I2CCLK periods)

To bridge the undefined region of the SCL falling edge, the user must set $SDADEL[3:0]$ so as to fulfill the following condition:

$$\left\{t_{H(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}] \right\} / ([PRESC + 1] \times t_{I2CCLK}) \leq SDADEL$$

$$SDADEL \leq \left\{t_{H(max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}] \right\} / ([PRESC + 1] \times t_{I2CCLK})$$
Note: \( t_{AF\,(\text{min})} \) and \( t_{AF\,(\text{max})} \) are only part of the condition when the analog filter is enabled. Refer to the device datasheet for \( t_{AF} \) values.

The \( t_{\text{HD;DAT}} \) time can at maximum be 3.45 \( \mu \text{s} \) for Standard-mode, 0.9 \( \mu \text{s} \) for Fast-mode, and 0.45 \( \mu \text{s} \) for Fast-mode Plus. It must be lower than the maximum of \( t_{\text{VD;DAT}} \) by a transition time. This maximum must only be met if the device does not stretch the LOW period \( (t_{\text{LOW}}) \) of the SCL signal. When it stretches 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. The previous condition then becomes:

\[
S\text{DADEL} \leq (t_{\text{VD;DAT \,(\text{max})}} - t_{r \,(\text{max})} - t_{AF \,(\text{max})} - [(DNF + 4) \times t_{\text{I2CCLK}}]) / ((PRESC + 1) \times t_{\text{I2CCLK}})
\]

**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[3:0] value.

After \( t_{\text{SDADEL}} \), or after sending SDA output when the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is \( t_{\text{SCLDEL}} = (S\text{CLDEL} + 1) \times t_{\text{PRESC}} \), where \( t_{\text{PRESC}} = (\text{PRESC} + 1) \times t_{\text{I2CCLK}} \). \( t_{\text{SCLDEL}} \) impacts the setup time \( t_{\text{SU;DAT}} \).

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

\[
[(t_{r \,(\text{max})} + t_{\text{SU;DAT \,(\text{min})}}) / ((\text{PRESC} + 1) \times t_{\text{I2CCLK}})] - 1 \leq \text{SCLDEL}
\]

Refer to the following table for \( t_{r} \), \( t_{\text{HD;DAT}} \), \( t_{\text{VD;DAT}} \), and \( t_{\text{SU;DAT}} \) standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

**Note:** At every clock pulse, after SCL falling edge detection, I2C operating as master or slave stretches SCL low during at least \( [SD\text{ADEL} + S\text{CLDEL} + 1] \times (\text{PRESC} + 1) \times t_{\text{I2CCLK}} \) in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps 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.

When the NOSTRETCH bit is set in slave mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

### Table 190. I²C-SMBus specification data setup and hold times

<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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>t_{\text{HD;DAT}}</td>
<td>Data hold time</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>t_{\text{VD;DAT}}</td>
<td>Data valid time</td>
<td>-</td>
<td>3.45</td>
<td>-</td>
<td>0.9</td>
</tr>
<tr>
<td>t_{\text{SU;DAT}}</td>
<td>Data setup time</td>
<td>250</td>
<td>-</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>t_{r}</td>
<td>Rise time of both SDA and SCL signals</td>
<td>-</td>
<td>1000</td>
<td>-</td>
<td>300</td>
</tr>
<tr>
<td>t_{r}</td>
<td>Fall time of both SDA and SCL signals</td>
<td>-</td>
<td>300</td>
<td>-</td>
<td>300</td>
</tr>
</tbody>
</table>
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] bitfields of the I2C_TIMINGR register.

When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. The $t_{SCLL}$ delay impacts the SCL low time $t_{LOW}$.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. The $t_{SCLH}$ impacts the SCL high time $t_{HIGH}$.

Refer to **I2C master initialization** for more details.

**Caution:** Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the slave NOSTRETCH settings must also be done before enabling the peripheral. Refer to **I2C slave initialization** for more details.

### Figure 293. I2C initialization flow

![I2C initialization flow diagram](image)

#### 32.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR
PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:
1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

### 32.4.7 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 eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and ninth SCL pulse (before the acknowledge pulse).

**Figure 294. Data reception**

![Figure 294. Data reception](image-url)
Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 295. Data transmission

Hardware transfer management

The I2C features an embedded byte counter 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

In master mode, the byte counter is always used. By default, it is disabled in slave mode. It can be enabled by software, by setting the SBC (slave byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number 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 of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.
When RELOAD = 0 in master mode, the counter can be used in two modes:

**Automatic end** (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] bitfield is transferred.

**Software end** (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] bitfield is 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 of the I2C_CR2 register is set. 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>
<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, all received bytes ACKed</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Slave Rx with ACK control</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

### 32.4.8 I2C slave mode

**I2C slave initialization**

To work in slave mode, the user must enable at least one slave address. The I2C_OAR1 and I2C_OAR2 registers are available to program the slave own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional slave addresses are required, the second slave address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] 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. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral 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 required, to perform software actions. If the master does not
support clock stretching, 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] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

**Slave with clock stretching**

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus slave stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled slave addresses.
- The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
- The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
- The stretch is released when I2C_RXDR is read.
- In slave byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
- The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
- The stretch is released after \([(SDADEL + SCLDEL + 1) \times (PRESC+1) + 1] \times t_{I2CCLK}\) period.

**Slave without clock stretching**

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus 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 of the I2C_CR1 register is set. 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, it 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 ninth 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 of the I2C_CR1 register is set.
**Slave byte control mode**

To allow byte ACK control in slave reception mode, the slave byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] 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 eighth and ninth 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 of 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 the next byte can be received.

NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

**Note:** The SBC bit must be configured when the I2C peripheral is disabled, 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:** The slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

*Figure 296. Slave initialization flow*

1. SBC must be set to support SMBus features.
Slave transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The slave automatically releases the SCL and SDA lines to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set 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 transmit must be programmed in NBYTES[7:0] 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[7:0].

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 to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees 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 (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.
Figure 297. Transfer sequence flow 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

I2C_ISR.TXIS = 1?

Yes

Write I2C_TXDR.TXDATA

SCL stretched
Figure 298. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1

Slave initialization

Slave transmission

I2C_ISR.TXIS = 1?

Yes

Write I2C_TXDR.TXDATA

No

I2C_ISR.TXIS = 1?

Yes

Set I2C_ICR.STOPCF

No

I2C_ISR.STOPF = 1?

Yes

Optional: Set I2C_ISR.TXE = 1 and I2C_ISR.TXIS=1
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

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. 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 300. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 0

Slave reception

Slave initialization

I2C_ISR.ADDR = 1?

No

Yes

Read ADDCODE and DIR in I2C_ISR
Set I2C_ICR.ADDRCF

I2C_ISR.RXNE = 1?

No

Yes

Write I2C_RXDR.RXDATA

SCL stretched
Figure 301. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 1

Figure 302. Transfer bus diagrams for I2C slave receiver (mandatory events only)

Example I2C slave receiver 3 bytes, NOSTRETCH = 0:

Example I2C slave receiver 3 bytes, NOSTRETCH = 1:

Legend

Transmission
Reception
SCL stretch

Legend

Transmission
Reception
SCL stretch
32.4.9 **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.

I2C detects its own SCL low level after a $t_{SYNC1}$ delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a $t_{SYNC2}$ delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + [(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}$$

The duration of $t_{SYNC1}$ depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): $D_{NF}[3:0] \times t_{I2CCLK}$
- delay due to SCL synchronization with the I2CCLK clock (two to three I2CCLK periods)

The duration of $t_{SYNC2}$ depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): $D_{NF}[3:0] \times t_{I2CCLK}$
- delay due to SCL synchronization with the I2CCLK clock (two to three I2CCLK periods)
Figure 303. Master clock generation

SCL master clock generation

- SCL high level detected
  - SCLH counter starts

SCL released

SCL driven low

SCL master clock synchronization

- SCL high level detected
  - SCLH counter starts

SCL driven low by another device

SCL low level detected
  - SCLL counter starts

SCL released

SCL low level detected
  - SCLL counter starts

SCL driven low by another device
Caution: For compliance with the I²C-bus or SMBus specification, the master clock must respect the timings in the following table.

### Table 192. 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></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>( t_{SCL} )</td>
<td>SCL clock frequency</td>
<td>-</td>
<td>100</td>
<td>-</td>
<td>400</td>
<td>-</td>
</tr>
<tr>
<td>( t_{HD:STA} )</td>
<td>Hold time (repeated) START condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>( t_{SU:STA} )</td>
<td>Set-up time for a repeated START condition</td>
<td>4.7</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>( t_{SU:STO} )</td>
<td>Set-up time for STOP condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>( t_{BUF} )</td>
<td>Bus free time between a STOP and START condition</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>( t_{LOW} )</td>
<td>Low period of the SCL clock</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>( t_{HIGH} )</td>
<td>High period of the SCL clock</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Note: The SCL[7:0] bitfield also determines the \( t_{BUF} \) and \( t_{SU:STA} \) timings and SCLH[7:0] the \( t_{HD:STA} \) and \( t_{SU:STO} \) timings.

Refer to Section 32.4.10 for examples of I2C_TIMINGR settings versus the I2CCLK frequency.

**Master communication initialization (address phase)**

To initiate the communication with a slave to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: slave address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.
To launch the communication, set the START bit of the I2C_CR2 register. The master then automatically sends a START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after the $t_{BUF}$ delay from a previous STOP condition expires.

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 is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the master automatically keeps resending the slave address in a loop until the first address byte (first seven address bits) is acknowledged by the slave. Setting the ADDRCF bit makes I2C quit that loop.

If the I2C peripheral is addressed as a slave (ADDR = 1) while the START bit is set, the I2C peripheral 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.

### 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 of 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 second byte + (Re)START + Slave address 10-bit header Read.

### Figure 304. Master initialization flow

![Master initialization flow diagram](image)

### Figure 305. 10-bit address read access with HEAD10R = 0

![10-bit address read access diagram](image)
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 306. 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
ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is
cleared when the I2C_TXDR register is written with the next data byte to transmit.

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 transmit is greater than 255, the reload
mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case,
when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL
line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched
  low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2
    register with the proper slave address configuration and the number of bytes to
    transfer. Setting the START bit clears the TC flag and sends the START condition
    on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2
    register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically
sent. the NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE
bit is set.
Figure 307. Transfer sequence flow for I2C master transmitter, $N \leq 255$ bytes
Figure 308. Transfer sequence flow for I2C master transmitter, 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

Set I2C_CR2.START with slave address
NBYTES ...

I2C_ISR.TXIS = 1?
Yes
Write I2C_TXDR

No

NBYTES transmitted ?
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

I2C_ISR.TCR = 1?

End
Figure 309. Transfer bus diagrams for I2C master transmitter
(mandatory events only)

Example I2C master transmitter 2 bytes, automatic end mode (STOP)

<table>
<thead>
<tr>
<th>S</th>
<th>Address</th>
<th>A</th>
<th>data1</th>
<th>A</th>
<th>data2</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>TXIS</td>
<td>TXIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EV1</td>
<td>EV2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- Transmission
- Reception
- SCL stretch

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)

<table>
<thead>
<tr>
<th>S</th>
<th>Address</th>
<th>A</th>
<th>data1</th>
<th>A</th>
<th>data2</th>
<th>ReS</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>TXIS</td>
<td>TXIS</td>
<td>TC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EV1</td>
<td>EV2</td>
<td>EV3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- Transmission
- Reception
- SCL stretch

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

MS19862V2
Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is 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 of the I2C_CR2 register, with the proper slave address configuration and the number of bytes to transfer. 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 of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.
Figure 310. Transfer sequence flow for I2C master receiver, N ≤ 255 bytes

Master initialization

NBYTES = N
AUTOEND = 0 for RESTART, 1 for STOP
Configure slave address
Set I2C_CR2.START

Yes

I2C_ISR.RXNE = 1?

No

Yes

Read I2C_RXDR

NBYTES received?

No

Yes

I2C_ISR.TC = 1?

No

Set I2C_CR2.START with slave address NBYTES...

Yes

End
Figure 311. Transfer sequence flow for I2C master receiver, N > 255 bytes

Master initialization

NBYTES = 0xFF; N=N-255
RELOAD = 1
Configure slave address
Set I2C_CR2.START

12C_ISR.RXNE = 1?
Yes
Read I2C_RXDR

NBYTES received?
Yes
Set I2C_CR2.START with slave address NBYTES ...

12C_ISR.TC = 1?
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

No
Yes

End
32.4.10 **I2C_TIMINGR register configuration examples**

The following tables provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

---

**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
### Table 193. Timing settings for \( f_{I^2CCLK} \) of 8 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>[\text{PRESC}[3:0]]</td>
<td>[0x1]</td>
<td>[0x1]</td>
<td>[0x0]</td>
</tr>
<tr>
<td>[\text{SCLL}[7:0]]</td>
<td>[0xC7]</td>
<td>[0x13]</td>
<td>[0x9]</td>
</tr>
<tr>
<td>[t_{SCL}]</td>
<td>[200 \times 250 \text{ ns} = 50 \mu s]</td>
<td>[20 \times 250 \text{ ns} = 5.0 \mu s]</td>
<td>[10 \times 125 \text{ ns} = 1250 \text{ ns}]</td>
</tr>
<tr>
<td>[\text{SCLH}[7:0]]</td>
<td>[0xC3]</td>
<td>[0xF]</td>
<td>[0x3]</td>
</tr>
<tr>
<td>[t_{SCLH}]</td>
<td>[196 \times 250 \text{ ns} = 49 \mu s]</td>
<td>[16 \times 250 \text{ ns} = 4.0 \mu s]</td>
<td>[4 \times 125 \text{ ns} = 500 \text{ ns}]</td>
</tr>
<tr>
<td>[t_{SCL}(1)]</td>
<td>[\sim 100 \mu s]</td>
<td>[\sim 10 \mu s]</td>
<td>[\sim 2.5 \mu s]</td>
</tr>
<tr>
<td>[\text{SDADEL}[3:0]]</td>
<td>[0x2]</td>
<td>[0x2]</td>
<td>[0x1]</td>
</tr>
<tr>
<td>[t_{SDADEL}]</td>
<td>[2 \times 250 \text{ ns} = 500 \text{ ns}]</td>
<td>[2 \times 250 \text{ ns} = 500 \text{ ns}]</td>
<td>[1 \times 125 \text{ ns} = 125 \text{ ns}]</td>
</tr>
<tr>
<td>[\text{SCLDEL}[3:0]]</td>
<td>[0x4]</td>
<td>[0x4]</td>
<td>[0x3]</td>
</tr>
<tr>
<td>[t_{SCLDEL}]</td>
<td>[5 \times 250 \text{ ns} = 1250 \text{ ns}]</td>
<td>[5 \times 250 \text{ ns} = 1250 \text{ ns}]</td>
<td>[4 \times 125 \text{ ns} = 500 \text{ ns}]</td>
</tr>
</tbody>
</table>

1. \(t_{SCL}\) 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 \(4 \times t_{I^2CCLK} = 500 \text{ ns}\). Example with \(t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}\).
3. \(t_{SYNC1} + t_{SYNC2}\) minimum value is \(4 \times t_{I^2CCLK} = 500 \text{ ns}\). Example with \(t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}\).
4. \(t_{SYNC1} + t_{SYNC2}\) minimum value is \(4 \times t_{I^2CCLK} = 500 \text{ ns}\). Example with \(t_{SYNC1} + t_{SYNC2} = 655 \text{ ns}\).

### Table 194. Timing settings for \( f_{I^2CCLK} \) of 16 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>[\text{PRESC}[3:0]]</td>
<td>[0x3]</td>
<td>[0x3]</td>
<td>[0x1]</td>
</tr>
<tr>
<td>[\text{SCLL}[7:0]]</td>
<td>[0xC7]</td>
<td>[0x13]</td>
<td>[0x9]</td>
</tr>
<tr>
<td>[t_{SCL}]</td>
<td>[200 \times 250 \text{ ns} = 50 \mu s]</td>
<td>[20 \times 250 \text{ ns} = 5.0 \mu s]</td>
<td>[10 \times 125 \text{ ns} = 1250 \text{ ns}]</td>
</tr>
<tr>
<td>[\text{SCLH}[7:0]]</td>
<td>[0xC3]</td>
<td>[0xF]</td>
<td>[0x3]</td>
</tr>
<tr>
<td>[t_{SCLH}]</td>
<td>[196 \times 250 \text{ ns} = 49 \mu s]</td>
<td>[16 \times 250 \text{ ns} = 4.0 \mu s]</td>
<td>[4 \times 125 \text{ ns} = 500 \text{ ns}]</td>
</tr>
<tr>
<td>[t_{SCL}(1)]</td>
<td>[\sim 100 \mu s]</td>
<td>[\sim 10 \mu s]</td>
<td>[\sim 2.5 \mu s]</td>
</tr>
<tr>
<td>[\text{SDADEL}[3:0]]</td>
<td>[0x2]</td>
<td>[0x2]</td>
<td>[0x2]</td>
</tr>
<tr>
<td>[t_{SDADEL}]</td>
<td>[2 \times 250 \text{ ns} = 500 \text{ ns}]</td>
<td>[2 \times 250 \text{ ns} = 500 \text{ ns}]</td>
<td>[2 \times 125 \text{ ns} = 250 \text{ ns}]</td>
</tr>
<tr>
<td>[\text{SCLDEL}[3:0]]</td>
<td>[0x4]</td>
<td>[0x4]</td>
<td>[0x3]</td>
</tr>
<tr>
<td>[t_{SCLDEL}]</td>
<td>[5 \times 250 \text{ ns} = 1250 \text{ ns}]</td>
<td>[5 \times 250 \text{ ns} = 1250 \text{ ns}]</td>
<td>[4 \times 125 \text{ ns} = 500 \text{ ns}]</td>
</tr>
</tbody>
</table>

1. \(t_{SCL}\) 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 \(4 \times t_{I^2CCLK} = 250 \text{ ns}\). Example with \(t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}\).
3. \(t_{SYNC1} + t_{SYNC2}\) minimum value is \(4 \times t_{I^2CCLK} = 250 \text{ ns}\). Example with \(t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}\).
4. \(t_{SYNC1} + t_{SYNC2}\) minimum value is \(4 \times t_{I^2CCLK} = 250 \text{ ns}\). Example with \(t_{SYNC1} + t_{SYNC2} = 500 \text{ ns}\).
32.4.11 SMBus specific features

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 operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Slave** is a device that receives or responds to a command.
- **Master** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized master that provides the main interface to the system CPU. A host must be a master-slave and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a master or a slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device can use any or all of them 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*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the 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 the SBC bit of the I2C_CR1 register. Refer to *Slave byte control mode* for more details.
Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then 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 I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus slave device can signal to the SMBus host 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/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus slave device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The 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 I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Description</th>
<th>Limits</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{\text{TIMEOUT}} )</td>
<td>Detect clock low timeout</td>
<td>25</td>
<td>35 ms</td>
</tr>
<tr>
<td>( t_{\text{LOW:SEXT}} )</td>
<td>Cumulative clock low extend time (slave device)</td>
<td>-</td>
<td>25 ms</td>
</tr>
<tr>
<td>( t_{\text{LOW:MEXT}} )</td>
<td>Cumulative clock low extend time (master device)</td>
<td>-</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

1. \( t_{\text{LOW:SEXT}} \) 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 \( t_{\text{LOW:SEXT}} \). Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. $t_{LOW:MEXT}$ 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 $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

### 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 $t_{IDLE} > t_{HIGH,MAX}$ (refer to [I2C timings](#)).

This timing parameter covers the condition where a master is 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 I2C peripheral supports a hardware bus idle detection.

### 32.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

**Received command and data acknowledge control (slave mode)**

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in slave mode, the slave byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to [Slave byte control mode](#) for more details.

**Specific addresses (slave mode)**

The specific SMBus addresses must be enabled if required. Refer to [Bus idle detection](#) for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.
The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

**Packet error checking**

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of 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 transferring NBYTES[7:0] - 1 data bytes, if 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 peripheral is enabled.

### Table 196. SMBus with PEC configuration

<table>
<thead>
<tr>
<th>Mode</th>
<th>SBC bit</th>
<th>RELOAD bit</th>
<th>AUTOEND bit</th>
<th>PECBYTE bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Tx/Rx NBYTES + PEC+ STOP</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Master Tx/Rx NBYTES + PEC + ReSTART</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Slave Tx/Rx with PEC</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

**Timeout detection**

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of 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.

**t_TIMEOUT check**

To check the t_TIMEOUT parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_I2CCLK period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to Table 197.

**Caution:** Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMOUTEN bit is set.

**t_LOW:SEXT and t_LOW:MEXT check**

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking t_LOW:SEXT for the I2C peripheral operating as a slave, or t_LOW:MEXT when it operates as a master. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_I2CCLK period, and within the timeout interval described in Bus idle detection section, the TIMEOUT flag of the I2C_ISR register is set.
Refer to Table 198.

**Caution:** Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

**Bus idle detection**

To check the T_IDLE period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMEOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_I2CCLK period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to Table 199.

**Caution:** Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

### 32.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach target t_TIMEOUT, t_LOW:SEXT, t_LOW:MEXT, and t_IDLE timings at different f_I2CCLK frequencies.

<table>
<thead>
<tr>
<th>f_I2CCLK</th>
<th>TIMEOUTA[11:0]</th>
<th>TIDLE</th>
<th>TIMEOUTEN</th>
<th>t_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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f_I2CCLK</th>
<th>TIMEOUTB[11:0]</th>
<th>TEXTEN</th>
<th>t_LOW:SEXT</th>
<th>t_LOW:MEXT</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>
<td></td>
</tr>
<tr>
<td>16 MHz</td>
<td>0x3F</td>
<td>1</td>
<td>64 x 2048 x 62.5 ns = 8 ms</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>f_I2CCLK</th>
<th>TIMEOUTA[11:0]</th>
<th>TIDLE</th>
<th>TIMEOUTEN</th>
<th>t_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>
</tbody>
</table>

### 32.4.14 SMBus slave mode

In addition to I2C slave transfer management (refer to Section 32.4.8: I2C slave mode), this section provides extra software flowcharts to support SMBus.

**SMBus slave transmitter**

When using the I2C peripheral in SMBus mode, set the SBC bit to enable 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. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 314. Transfer sequence flow for SMBus slave transmitter N bytes + PEC**
Figure 315. Transfer bus diagram 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 Address</td>
<td>A</td>
<td>data1</td>
</tr>
<tr>
<td>EV1</td>
<td>EV2</td>
<td>EV3</td>
</tr>
</tbody>
</table>

NBYTES 3

EV1: ADDR ISR: check ADDCODE, program NBYTES=3, set PECBYTE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2

SMBus slave receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to Slave byte control mode for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, 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.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 316. Transfer sequence flow for SMBus slave receiver N bytes + PEC

- **SMBus slave reception**
- **Slave initialization**

  - **I2C_ISR.ADDR = 1?**
    - **Yes**
      - Read ADDCODE and DIR in I2C_ISR
      - I2C_CR2.NBYTES = 1, RELOAD = 1
      - PECBYTE = 1
      - Set I2C_ICR.ADDRCF
    - **No**

  - **I2C_ISR.ADDR = 1?**
    - **Yes**
      - Read ADDCODE and DIR in I2C_ISR
      - I2C_CR2.NBYTES = 1, RELOAD = 1
      - PECBYTE = 1
      - Set I2C_ICR.ADDRCF
    - **No**

  - **I2C_ISR.RXNE = 1? I2C_ISR.TCR = 1?**
    - **Yes**
      - Read I2C_RXDR.RXDATA
      - Program I2C_CR2.NACK = 0
      - I2C_CR2.NBYTES = 1
      - N = N - 1
    - **No**

  - **N = 1?**
    - **Yes**
      - Read I2C_RXDR.RXDATA
      - Program RELOAD = 0
      - NACK = 0 and NBYTES = 1
    - **No**

  - **I2C_ISR.RXNE = 1?**
    - **Yes**
      - Read I2C_RXDR.RXDATA
    - **No**

- **End**
**32.4.15 SMBus master mode**

In addition to I2C master transfer management (refer to Section 32.4.9: I2C master mode), this section provides extra software flowcharts 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 loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, the 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, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. 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 318. 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, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the slave address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, 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, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the slave address 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.
**Wake-up from Stop mode on address match**

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI oscillator must be selected as the clock source for I2CCLK to allow the wake-up from Stop mode.

In Stop mode, the HSI oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI oscillator and stretches SCL low until the oscillator wakes up.

HSI is then used for the address reception.
If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI oscillator is stopped again and the device does not wake up.

**Note:** When the system clock is used as I2C clock, or when WUPEN = 0, the HSI oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a master or as an addressed slave after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

**Caution:** The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

**Caution:** The feature is only available when the HSI oscillator is selected as the I2C clock.

**Caution:** Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

**Caution:** If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

### 32.4.17 Error conditions

The following errors are the conditions that can cause the 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 nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as master or addressed slave (that is, not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

**Arbitration loss (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 of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.
**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 must 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 of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

**Packet error checking error (PECERR)**

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

**Timeout error (TIMEOUT)**

A timeout error occurs for any of these conditions:
- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus tLOW:MEXT parameter).
- Slave cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus tLOW: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 of the I2C_CR1 register is set.

**Alert (ALERT)**

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.
### 32.5 I2C in low-power modes

Table 200. Effect of low-power modes to I2C

<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 contents of I2C registers are kept.</td>
</tr>
<tr>
<td></td>
<td>– WUPEN = 1 and I2C is clocked by an internal oscillator (HSI). The address</td>
</tr>
<tr>
<td></td>
<td>recognition is functional. The I2C address match condition causes the</td>
</tr>
<tr>
<td></td>
<td>device to exit the Stop mode.</td>
</tr>
<tr>
<td></td>
<td>– WUPEN = 0: the I2C must be disabled before entering Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The I2C peripheral is powered down. It must be reinitialized after exiting</td>
</tr>
<tr>
<td></td>
<td>Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 32.3: I2C implementation for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

### 32.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 201. 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 Sleep mode</th>
<th>Exit Stop modes</th>
<th>Exit Standby modes</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>Yes</td>
<td>No</td>
<td>No</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>
<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></td>
<td></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>Yes</td>
<td>No</td>
<td>No</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>
<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>
<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>
<td></td>
</tr>
<tr>
<td>I2C_ER</td>
<td>Bus error</td>
<td>BERR</td>
<td>ERRIE</td>
<td>Write BERRCF = 1</td>
<td>Yes</td>
<td>No</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>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun/Underrun</td>
<td>OVR</td>
<td></td>
<td>Write OVRCF = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2C_ER</td>
<td>PEC error</td>
<td>PECERR</td>
<td>ERRIE</td>
<td>Write PECERRCF = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Timeout/(t)_LOW error</td>
<td>TIMEOUT</td>
<td></td>
<td>Write TIMEOUTCF = 1</td>
<td></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>
<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 wake-up from Stop mode feature. Refer to Section 32.3: I2C implementation.
32.7 I2C DMA requests

Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see Section 11: Direct memory access controller (DMA)) 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, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to Master transmitter.

In slave mode:
- With NOSTRETCH = 0, when all data are transferred using DMA, 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.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to SMBus slave transmitter and SMBus master mode.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to Section 11: Direct memory access controller (DMA)) 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, 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.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to SMBus slave receiver and SMBus master receiver.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

32.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.
### 32.9 I2C registers

Refer to Section 1.2 on page 61 for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

#### 32.9.1 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 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></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>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>RXDMA EN</th>
<th>TXDMA EN</th>
<th>Res.</th>
<th>ANF OFF</th>
<th>DNF[3:0]</th>
<th>ERRIE</th>
<th>TCIE</th>
<th>STOP IE</th>
<th>NACK IE</th>
<th>ADDR IE</th>
<th>RXIE</th>
<th>TXIE</th>
<th>PE</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>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

**Bit 23 PECEN:** PEC enable

0: PEC calculation disabled
1: PEC calculation enabled

**Bit 22 ALERTEN:** SMBus alert enable

0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

*Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.*

**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.

**Bit 20 SMBHEN:** SMBus host address enable

0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

**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:** Wake-up from Stop mode enable

0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.

*Note: WUPEN can be set only when DNF[3:0] = 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 be programmed only when the I2C peripheral 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 **ANFOFF**: Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled

*Note:* This bit can be programmed only when the I2C peripheral 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 $2^{DNF[3:0]} \times t_{I2CCLK}$
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to one $t_{I2CCLK}$
...
1111: Digital filter enabled and filtering capability up to fifteen $t_{I2CCLK}$

*Note:* If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral 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 generates 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 generates an interrupt:
- Transfer complete (TC)
- Transfer complete reload (TCR)
### 32.9.2 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 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>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
</table>
| 5   | STOPIE: STOP detection interrupt enable | 0: STOP detection (STOPF) interrupt disabled  
1: STOP detection (STOPF) interrupt enabled | |
| 4   | NACKIE: Not acknowledge received interrupt enable | 0: Not acknowledge (NACKF) received interrupts disabled  
1: Not acknowledge (NACKF) received interrupts enabled | |
| 3   | ADDRIE: Address match interrupt enable (slave only) | 0: Address match (ADDR) interrupts disabled  
1: Address match (ADDR) interrupts enabled | |
| 2   | RXIE: RX interrupt enable | 0: Receive (RXNE) interrupt disabled  
1: Receive (RXNE) interrupt enabled | |
| 1   | TXIE: TX interrupt enable | 0: Transmit (TXIS) interrupt disabled  
1: Transmit (TXIS) interrupt enabled | |
| 0   | PE: Peripheral enable | 0: Peripheral disabled  
1: Peripheral enabled | |

**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 three APB clock cycles.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
</table>
| 26  | PECBYTE: Packet error checking byte | 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, and in slave mode when SBC = 0.
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.

*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 condition generation
This bit only pertains to master mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.
0: No STOP generation
1: STOP generation after current byte transfer
*Note*: Writing 0 to this bit has no effect.

Bit 13  **START**: START condition generation
This bit is set by software. It is cleared by hardware after the START condition 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 setting the ADDRCF bit of 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 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.
1: The master sends only the first seven 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 9:0 **SADD[9:0]**: Slave address (master mode)

**Condition: In 7-bit addressing mode (ADD10 = 0):**

SADD[7:1] must be written with the 7-bit slave address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

**Condition: In 10-bit addressing mode (ADD10 = 1):**

SADD[9:0] must be written with the 10-bit slave address to be sent.

*Note: Changing these bits when the START bit is set is not allowed.*

### 32.9.3 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 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>Bit 31-16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th><strong>OA1EN</strong>: Own address 1 enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Own address 1 disabled. The received slave address OA1 is NACKed.</td>
</tr>
<tr>
<td>1</td>
<td>Own address 1 enabled. The received slave address OA1 is ACKed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 14:11</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 10</th>
<th><strong>OA1MODE</strong>: Own address 1 10-bit mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Own address 1 is a 7-bit address.</td>
</tr>
<tr>
<td>1</td>
<td>Own address 1 is a 10-bit address.</td>
</tr>
</tbody>
</table>

*Note: This bit can be written only when OA1EN = 0.*
32.9.4 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 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 2x PCLK1 + 6 x I2CCLK.

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15</td>
<td>OA2EN: Own address 2 enable</td>
</tr>
<tr>
<td></td>
<td>0: Own address 2 disabled. The received slave address OA2 is NACKed.</td>
</tr>
<tr>
<td></td>
<td>1: Own address 2 enabled. The received slave address OA2 is ACKed.</td>
</tr>
<tr>
<td>Bit 14:11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 10:8</td>
<td>OA2MSK[2:0]: Own address 2 masks</td>
</tr>
<tr>
<td></td>
<td>000: No mask</td>
</tr>
<tr>
<td></td>
<td>001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.</td>
</tr>
<tr>
<td></td>
<td>010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.</td>
</tr>
<tr>
<td></td>
<td>100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.</td>
</tr>
<tr>
<td></td>
<td>111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.</td>
</tr>
</tbody>
</table>

Note: These bits can be written only when OA2EN = 0.

As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

| Bit 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.
### 32.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>Address offset: 0x10</th>
<th>Reset value: 0x0000 0000</th>
<th>Access: no wait states</th>
</tr>
</thead>
</table>

#### Bits 31:28 PRESC[3:0]: Timing prescaler
This field is used to prescale I2CCLK to generate the clock period \( t_{\text{PRESC}} \) used for data setup and hold counters (refer to section [I2C timings](#)), and for SCL high and low level counters (refer to section [I2C master initialization](#)).

\[
t_{\text{PRESC}} = (\text{PRESC} + 1) \times t_{\text{I2CCLK}}
\]

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

#### 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 \( t_{\text{SCLDEL}} = (\text{SCLDEL} + 1) \times t_{\text{PRESC}} \) between SDA edge and SCL rising edge. In master and in slave modes with NOSTRETCH = 0, the SCL line is stretched low during \( t_{\text{SCLDEL}} \).

Note: \( t_{\text{SCLDEL}} \) is used to generate \( t_{\text{SU:DAT}} \) timing.

#### Bits 19:16 SDADEL[3:0]: Data hold time
This field is used to generate the delay \( t_{\text{SDADEL}} \) between SCL falling edge and SDA edge. In master and in slave modes with NOSTRETCH = 0, the SCL line is stretched low during \( t_{\text{SDADEL}} \).

\[
t_{\text{SDADEL}} = \text{SDADEL} \times t_{\text{PRESC}}
\]

Note: \( t_{\text{SDADEL}} \) is used to generate \( t_{\text{HD:DAT}} \) 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.

\[
t_{\text{SCLH}} = (\text{SCLH} + 1) \times t_{\text{PRESC}}
\]

Note: \( t_{\text{SCLH}} \) is also used to generate \( t_{\text{SU:STO}} \) and \( t_{\text{HD:STA}} \) 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.

\[
t_{\text{SCLL}} = (\text{SCLL} + 1) \times t_{\text{PRESC}}
\]

Note: \( t_{\text{SCLL}} \) is also used to generate \( t_{\text{BUF}} \) and \( t_{\text{SU:STA}} \) timings.

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.
### 32.9.6 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 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></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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIMOUTEN</th>
<th>Res.</th>
<th>Res.</th>
<th>TIDLE</th>
<th>TIMEOUTA[11:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

- **Bit 31** **TEXTEN**: Extended clock timeout enable  
  0: Extended clock timeout detection is disabled  
  1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than t\_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.

- **Bits 27:16** **TIMEOUTB[11:0]**: Bus timeout B  
  This field is used to configure the cumulative clock extension timeout:  
  - Master mode: the master cumulative clock low extend time (t\_LOW\_MEXT) is detected  
  - Slave mode: the slave cumulative clock low extend time (t\_LOW\_SEXT) is detected  
  
  \[ t\_LOW\_EXT = (\text{TIMEOUTB} + \text{TIDLE} = 01) \times 2048 \times t\_I2CCLK \]  
  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\_TIMEOUT (TIDLE = 0) or high for more than t\_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\_TIMEOUT when TIDLE = 0  
  \[ t\_TIMEOUT = (\text{TIMEOUTA} + 1) \times 2048 \times t\_I2CCLK \]  
  - The bus idle condition (both SCL and SDA high) when TIDLE = 1  
  \[ t\_IDLE = (\text{TIMEOUTA} + 1) \times 4 \times t\_I2CCLK \]  
  Note: These bits can be written only when TIMOUTEN = 0.
32.9.7  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>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>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>
<td>rs</td>
<td>s</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 two 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, and 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 an 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.

Bit 12  TIMEOUT: Timeout or tLOW 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.

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.

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 the 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.*

Bit 6 **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:
- as a master, provided that the STOP condition is generated by the peripheral.
- 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 5 **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 4 **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.*

Bit 3 **RXNE**: Receive data register not empty (receivers)
This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 2 **TXIS**: Transmit interrupt status (transmitters)
This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

*Note: This bit is cleared by hardware when PE = 0.*
Bit 0  **TXE**: Transmit data register empty (transmitters)
  This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next
data to be sent is written in the I2C_TXDR register.
  This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.
  *Note:* This bit is set by hardware when PE = 0.

---

## 32.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

<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>ALERTCF: Alert flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>14</td>
<td>TIMOUTCF: Timeout detection flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>13</td>
<td>PECCF: PEC error flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>12</td>
<td>OVRCF: Overrun/underrun flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the OVR flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>11</td>
<td>ARLOCF: Arbitration lost flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>10</td>
<td>BERRCF: Bus error flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.</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>STOPCF: STOP detection flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>4</td>
<td>NACKCF: Not acknowledge flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the NACKF flag in I2C_ISR register.</td>
</tr>
<tr>
<td>3</td>
<td>ADDRCF: Address matched flag clear</td>
</tr>
<tr>
<td></td>
<td>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.</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>
### 32.9.9 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>

*PEC*[7:0]*

<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 31:8** 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.

### 32.9.10 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>

*RXDATA*[7:0]*

<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 31:8** Reserved, must be kept at reset value.
- **Bits 7:0** *RXDATA*[7:0]: 8-bit receive data
  - Data byte received from the I²C-bus.
32.9.11   I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

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.
### 32.9.12 I2C register map

The table below provides the I2C register map and the reset values.

#### Table 202. I2C register map and reset values

| Offset | Register name | 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 | Reset value |
|--------|---------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-------|
| 0x00   | I2C_CR1       | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x04   | I2C_CR2       | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x08   | I2C_OAR1      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x0C   | I2C_OAR2      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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   | I2C_TIMINGR   | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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   | I2C_TIMEOUTR  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x18   | I2C_ISR       | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x1C   | I2C_ICR       | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x20   | I2C_PECR      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x24   | I2C_RXDR      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |
| 0x28   | I2C_TXDR      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 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 |

**Reset value**

- 0x00: I2C_CR1
- 0x04: I2C_CR2
- 0x08: I2C_OAR1
- 0x0C: I2C_OAR2
- 0x10: I2C_TIMINGR
- 0x14: I2C_TIMEOUTR
- 0x18: I2C_ISR
- 0x1C: I2C_ICR
- 0x20: I2C_PECR
- 0x24: I2C_RXDR
- 0x28: I2C_TXDR
Refer to *Section 2.2 on page 66* for the register boundary addresses.
Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

33.1 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.
33.2 **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: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
33.3 **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

33.4 **USART implementation**

The table(s) below describe(s) USART implementation. It(they) also include(s) LPUART for comparison.

<table>
<thead>
<tr>
<th>USART / LPUART modes/features(^{(1)})</th>
<th>USART1</th>
<th>LPUART1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire half-duplex communication</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain and wake-up from low-power mode</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>USART data length</td>
<td></td>
<td>7, 8 and 9 bits</td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

1. X = supported.
33.5 **USART functional description**

33.5.1 **USART block diagram**

The simplified block diagram given in Figure 320 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 CK 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.
33.5.2 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 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.**

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.

Refer to Table 204 and Table 205 for the list of USART input/output pins and internal signals.
### Description of USART input/output signals

#### Table 204. USART/UART input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART_RX/UART_RX</td>
<td>Input</td>
<td>Serial data receive input</td>
</tr>
<tr>
<td>USART_TX/UART_TX</td>
<td>Output</td>
<td>Transmit data output</td>
</tr>
<tr>
<td>USART_CTS/UART_CTS</td>
<td>Input</td>
<td>Clear to send</td>
</tr>
<tr>
<td>USART_RTS/UART_RTS</td>
<td>Output</td>
<td>Request to send</td>
</tr>
<tr>
<td>USART_De(1)/UART_De(2)</td>
<td>Output</td>
<td>Driver enable</td>
</tr>
<tr>
<td>USART_CK</td>
<td>Output</td>
<td>Clock output in synchronous master and smartcard modes.</td>
</tr>
<tr>
<td>USART_NSS(3)</td>
<td>Input</td>
<td>Slave select input in synchronous slave mode.</td>
</tr>
</tbody>
</table>

1. USART_De and USART_RTS share the same pin.
2. UART_De and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

#### Table 205. USART internal input/output signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usart_pclk</td>
<td>Input</td>
<td>APB clock</td>
</tr>
<tr>
<td>usart_ker_ck</td>
<td>Input</td>
<td>USART kernel clock</td>
</tr>
<tr>
<td>usart_wkup</td>
<td>Output</td>
<td>USART provides a wake-up interrupt</td>
</tr>
<tr>
<td>usart_it</td>
<td>Output</td>
<td>USART global interrupt</td>
</tr>
<tr>
<td>usart_tx_dma</td>
<td>Input/output</td>
<td>USART transmit DMA request</td>
</tr>
<tr>
<td>usart_rx_dma</td>
<td>Input/output</td>
<td>USART receive DMA request</td>
</tr>
</tbody>
</table>

### 33.5.3 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 321):

- 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).
**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 321. Word length programming**

- **9-bit word length (M = 01), 1 Stop bit**
- **8-bit word length (M = 00), 1 Stop bit**
- **7-bit word length (M = 10), 1 Stop bit**

**LBCL bit controls last data clock pulse**

MS33194V2
33.5.4 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 is 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 is 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.

33.5.5 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 CK 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 is supported by normal USART, single-wire and modem modes.
- **1.5 stop bits:** To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission features 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 322**). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 322. Configurable stop bits**
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 33.5.19: Continuous communication using USART and DMA.
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 device to enter the low-power mode (see *Figure 323: TC/TXE behavior when transmitting*).
**Figure 323. 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 321).

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 TXFIFO is full.

**Idle characters**

Setting the TE bit drives the USART to send an idle frame before the first data frame.

### 33.5.6 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 0X 0X 0 X 0X 0.
Figure 324. Start bit detection when oversampling by 16 or 8

<table>
<thead>
<tr>
<th>RX state</th>
<th>Idle</th>
<th>Start bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal sample clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real sample clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditions to validate the start bit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 33.5.19: Continuous communication using USART and DMA.
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 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 is not lost. The previous data is available by reading the USART_RDR register.
  
  - the shift register is 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 is not lost. It is available by reading the USART_RDR register.
  
  - The shift register is 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 Section Reset and clock control (RCC)). 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 wake-up from low-power mode features are supported, the `usart_ker_ck` clock source can be configurable in the RCC (see Section Reset and clock control (RCC)). 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_PRES` register.

Some `usart_ker_ck` sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up 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 enable USART communications.

The communication speed range (especially 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 enables 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 326 and Figure 327).

Depending on your application:
   - select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to `usart_ker_ck_pres/8`). In this case the maximum receiver tolerance to clock deviation is reduced (refer to Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1055)
   - 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
usrart_ker_ck_pres/16 (where usrart_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 206) 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 33.5.8: Tolerance of the USART receiver to clock deviation on page 1055). In this case the NE bit is never 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 is 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 326. Data sampling when oversampling by 16](image)
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 is 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 33.5.16: USART receiver timeout on page 1069 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.

33.5.7 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:

\[
\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ckpres}}{\text{USARTDIV}}
\]

In case of oversampling by 8, the baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = 2 \times \frac{\text{usart\_ker\_ckpres}}{\text{USARTDIV}}
\]

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ckpres}}{\text{USARTDIV}}
\]
USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

When \( \text{OVER8} = 0 \), \( \text{BRR} = \text{USARTDIV} \).

When \( \text{OVER8} = 1 \)
  - \( \text{BRR}[2:0] = \text{USARTDIV}[3:0] \) shifted 1 bit to the right.
  - \( \text{BRR}[3] \) must be kept cleared.
  - \( \text{BRR}[15:4] = \text{USARTDIV}[15:4] \)

**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, \( \text{USARTDIV} \) must be greater than or equal to 16.

**How to derive USARTDIV from USART_BRR register values**

**Example 1**

To obtain 9600 baud with \( \text{usart_ker_ck_pres} = 8 \) MHz:

In case of oversampling by 16:
- \( \text{USARTDIV} = 8 \, 000 \, 000/9600 \)
- \( \text{BRR} = \text{USARTDIV} = 0d833 = 0x0341 \)

In case of oversampling by 8:
- \( \text{USARTDIV} = 2 \times 8 \, 000 \, 000/9600 \)
- \( \text{USARTDIV} = 1666,66 \) (0d1667 = 0x683)
- \( \text{BRR}[3:0] = 0x3 >> 1 = 0x1 \)
- \( \text{BRR} = 0x681 \)

**Example 2**

To obtain 921.6 Kbaud with \( \text{usart_ker_ck_pres} = 48 \) MHz:

In case of oversampling by 16:
- \( \text{USARTDIV} = 48 \, 000 \, 000/921 \, 600 \)
- \( \text{BRR} = \text{USARTDIV} = 0d52 = 0x34 \)

In case of oversampling by 8:
- \( \text{USARTDIV} = 2 \times 48 \, 000 \, 000/921 \, 600 \)
- \( \text{USARTDIV} = 104 \) (0d104 = 0x68)
- \( \text{BRR}[3:0] = \text{USARTDIV}[3:0] >> 1 = 0x8 >> 1 = 0x4 \)
- \( \text{BRR} = 0x64 \)

**33.5.8 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)

\[
\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} \quad \text{USART receiver tolerance}
\]

where \(\text{DWU}\) is the error due to sampling point deviation when the wake-up from low-power mode is used.

When \(M[1:0] = 01\):

\[
\text{DWU} = \frac{t_{\text{WU USART}}}{11} \text{Tbit}
\]

When \(M[1:0] = 00\):

\[
\text{DWU} = \frac{t_{\text{WU USART}}}{10} \text{Tbit}
\]

When \(M[1:0] = 10\):

\[
\text{DWU} = \frac{t_{\text{WU USART}}}{9} \text{Tbit}
\]

\(t_{\text{WU USART}}\) 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.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 207, Table 208, 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 207. Tolerance of the USART receiver when BRR [3:0] = 0000

<table>
<thead>
<tr>
<th>M bits</th>
<th>(\text{OVER8 bit} = 0)</th>
<th>(\text{OVER8 bit} = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\text{ONEBIT} = 0)</td>
<td>(\text{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>
Note: The data specified in Table 207 and Table 208 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).

33.5.9 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 enables 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 then waits 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 is 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.
33.5.10 **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 328.*
If the MMRQ is set while the IDLE character has already elapsed, mute mode is not 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 not 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 329.
33.5.11 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 USART_CR2 register and the RTOIE in the USART_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.
33.5.12 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 209.

Table 209. USART frame formats

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>USART 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>
<tr>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

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, the parity bit is equal to 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 is equal to 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).
33.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to Section 33.4: USART implementation on page 1038.

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:
- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in Section 33.5.4 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 two bits of value ‘1’ are sent to enable 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 is 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 330: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 1064.

Examples of break frames are given on Figure 331: Break detection in LIN mode vs. Framing error detection on page 1065.
Figure 330. 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

Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 Idle

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

Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 Idle

Read samples 0 0 0 0 0 0 0 0 0 0

LBDF

Delimiter is immediate

Case 3: break signal long enough => break detected, LBDF is set

RX line

Capture strobe

Break state machine

Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 wait delimiter Idle

Read samples 0 0 0 0 0 0 0 0 0 0

LBDF

wait delimiter
33.5.14 **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 CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK 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 332, Figure 333 and Figure 334).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK 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 CK (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 CK 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 332. USART example of synchronous master transmission

Figure 333. USART data clock timing diagram in synchronous master mode (M bits = 00)
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 CK 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 335).

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)**
  - The SPI slave is always selected and NSS input pin is 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 becomes 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 transmits 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 enables 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 CK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 $\text{usart.ker.ck}$ cycles before the first CK edge.

### 33.5.15 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.

### 33.5.16 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 is generated if RTOIE bit in USART_CR1 register is set.
33.5.17 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to Section 33.4: USART implementation on page 1038.

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: \( \text{STOP} = '11' \) 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 336* shows examples of what can be seen on the data line with and without parity error.

*Figure 336. ISO 7816-3 asynchronous protocol*

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 deassertion 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 337 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 CK output. In smartcard mode, CK 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. CK frequency can be programmed from \( \frac{\text{usart_ker_ck_pres}}{2} \) to \( \frac{\text{usart_ker_ck_pres}}{62} \), 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 is 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 enable 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 signals 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.

---

**Figure 337. Parity error detection using the 1.5 stop bits**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Parity bit</th>
<th>1.5 Stop bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit time</td>
<td>Sampling at 8th, 9th, 10th</td>
<td>1.5 bit time</td>
</tr>
<tr>
<td>0.5 bit time</td>
<td>Sampling at 8th, 9th, 10th</td>
<td>0.5 bit time</td>
</tr>
<tr>
<td>0.5 bit time</td>
<td>Sampling at 8th, 9th, 10th</td>
<td>0.5 bit time</td>
</tr>
</tbody>
</table>

---

MSv31163V1
(0x0). 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 is 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, assuming that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH results in a USART received character of 03 and an odd parity.
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 is 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.

### 33.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to Section 33.4: USART implementation on page 1038.

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 CKLEN 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 338).

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 kbaud 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 339).

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 two periods are 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 338. IrDA SIR ENDEC block diagram

Figure 339. IrDA data modulation (3/16) - normal mode
33.5.19 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 33.4: USART implementation on page 1038 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 33.5.6. 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.
**Figure 340. Transmission using DMA**

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX line</td>
<td>Set by hardware cleared by DMA read</td>
<td>Set by hardware cleared by DMA read</td>
</tr>
<tr>
<td>DMA request</td>
<td>Ignored by the DMA because the transfer is complete</td>
<td></td>
</tr>
<tr>
<td>TXE flag</td>
<td>Set by hardware</td>
<td>Set by hardware</td>
</tr>
<tr>
<td>DMA writes USART TDR</td>
<td>DMA writes F1 into USART_TDR</td>
<td>DMA writes F2 into USART_TDR</td>
</tr>
<tr>
<td>DMA TCIF flag (transfer complete)</td>
<td>Cleared by software</td>
<td>ignored by the DMA because the transfer is complete</td>
</tr>
<tr>
<td>Software configures DMA to send 3 data blocks and enables USART</td>
<td>Software writes DMA TCIF flag (transfer complete)</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 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.
Figure 341. Reception 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>RXNE flag</td>
<td>Set by hardware</td>
<td>cleared by DMA read</td>
<td></td>
</tr>
<tr>
<td>DMA request</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USART RDR F1</td>
<td>DMA reads F1 from USART RDR</td>
<td>DMA reads F2 from USART RDR</td>
<td>DMA reads F3 from USART RDR</td>
</tr>
<tr>
<td>DMA TCIF flag (transfer complete)</td>
<td></td>
<td></td>
<td>The DMA transfer is complete (TCIF=1 in DMA_ISR)</td>
</tr>
</tbody>
</table>

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 USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

33.5.20 RS232 hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The Figure 342 shows how to connect 2 devices in this mode:

Figure 342. Hardware flow control between 2 USARTs
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 RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 343* shows an example of communication with RTS flow control enabled.

![Figure 343. RS232 RTS flow control](image)

*Note:* When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (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 CTS is asserted 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 CTS 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 344* shows an example of communication with CTS flow control enabled.
RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables 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 deassertion 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).

Note: For correct behavior, CTS must be deasserted 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.
33.5.21 **USART low-power management**

The USART has advanced low-power mode functions, that enables 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 wake-up 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 enables 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
  wake-up 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 wake up the MCU from low-power mode enables 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 wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup**
interrupt is generated if the WUFIE bit is set.
**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 wake-up 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:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.

  - If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains 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 wake-up 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).

**Wake-up 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 wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up 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 wake-up event programmed to “address match detection” and FIFO management disabled.
Figure 345 shows the USART behavior when the wake-up event is verified.

**Figure 345. Wake-up event verified (wake-up event = address match, FIFO disabled)**

![Figure 345](MSv40856V2)

Figure 346 shows the USART behavior when the wake-up event is not verified.

**Figure 346. Wake-up event not verified (wake-up event = address match, FIFO disabled)**

![Figure 346](MSv40857V2)

**Note:** The figures above are valid when address match or any received frame is used as wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.
Determining the maximum USART baud rate that enables to correctly wake up the device from low-power mode

The maximum baud rate that enables to correctly wake up the device from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see Section 33.5.8: 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 207: Tolerance of the USART receiver when BRR [3:0] = 0000, the USART receiver tolerance equals 3.41%.

\[
\text{DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance}
\]

\[
D_{\text{WUmax}} = \frac{t_{\text{WUUSART}}}{11 \times T_{\text{bit Min}}}
\]

\[
T_{\text{bit Min}} = \frac{t_{\text{WUUSART}}}{11 \times D_{\text{WUmax}}}
\]

where \( t_{\text{WUUSART}} \) is the wake-up 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:

\[
t_{\text{WUUSART}} = 3 \mu\text{s (values provided only as examples; for correct values, refer to the device datasheet)}.
\]

\[
D_{\text{WUmax}} = 3.41\% - 1\% = 2.41\%
\]

\[
T_{\text{bit min}} = 3 \mu\text{s} / (11 \times 2.41\%) = 11.32 \mu\text{s}.
\]

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: \( 1/11.32 \mu\text{s} = 88.36 \text{ Kbaud} \).

### 33.6 USART in low-power modes

Table 210. Effect of low-power modes on the USART

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. USART interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^{(1)})</td>
<td>The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The USART peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 33.4: USART implementation to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.
### 33.7 USART interrupts

Refer to Table 211 for a detailed description of all USART interrupt requests.

**Table 211. USART interrupt requests**

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop(1) modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART or UART</td>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>Write TDR</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFIFO full</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>Write TDR or write 1 in TXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTEIE</td>
<td>Write TDR</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>Write 1 in CTSCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>Write TDR or write 1 in TCCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmission Complete Before Guard Time</td>
<td>TCBGT</td>
<td>TCBGTIE</td>
<td>Write TDR or write 1 in TCBGT</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>USART or UART</td>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>Read RDR or write 1 in RXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>Read RDR until RXFIFO empty or write 1 in RXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Receive FIFO Full</td>
<td>RXFF(2)</td>
<td>RXFFIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RXNEIE/ RXFNEIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>Write 1 in IDLECF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>Write 1 in PECF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>LIN break</td>
<td>LBDF</td>
<td>LBDFIE</td>
<td>Write 1 in LBDCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Noise error in multibuffer communication</td>
<td>NE</td>
<td>EIE</td>
<td>Write 1 in NFCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Overrun error in multibuffer communication</td>
<td>ORE(3)</td>
<td>EIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Framing Error in multibuffer communication</td>
<td>FE</td>
<td>EIE</td>
<td>Write 1 in FECF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>Write 1 in CMCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Receiver timeout</td>
<td>RTOF</td>
<td>RTOFIE</td>
<td>Write 1 in RTOCCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>End of Block</td>
<td>EOBF</td>
<td>EOBIE</td>
<td>Write 1 in EOBCCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Wake-up from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>Write 1 in WUC</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>SPI slave underrun error</td>
<td>UDR</td>
<td>EIE</td>
<td>Write 1 in UDRCF</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to Section 33.4: USART implementation for the list of supported Stop modes.
2. 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 is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

3. When OVRDIS = 0.

### 33.8 USART registers

Refer to Section 1.2 on page 61 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

#### 33.8.1 USART control register 1 (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>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RXFFIE</td>
<td>RXFIFO full interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Interrupt inhibited</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: USART interrupt generated when RXFF = 1 in the USART_ISR register</td>
</tr>
<tr>
<td>30</td>
<td>TXFEIE</td>
<td>TXFIFO empty interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Interrupt inhibited</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: USART interrupt generated when TXFE = 1 in the USART_ISR register</td>
</tr>
<tr>
<td>29</td>
<td>FIFOEN</td>
<td>FIFO mode enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bitfield can only be written when the USART is disabled (UE = 0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: 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>
</tbody>
</table>
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': 1 start bit, 8 Data bits, n Stop bit
- M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit
- M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit
This bit can only be written when the USART is disabled (UE = 0).
**Note:** In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

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 **MO**: 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 wake-up method
This bit determines the USART wake-up 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 is 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 wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.

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 CK is always available when CLKEN = 1, regardless of the UE bit value.
33.8.2 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>Bit</th>
<th>Description</th>
<th>Value</th>
<th>User Controls</th>
<th>Data Controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>FIFOEN: FIFO mode enable</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>M1: Word length</td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>EOBIE: End of Bblock interrupt enable</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Interrupt inhibited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART interrupt generated when the EOBF flag is set in the USART_ISR register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>RTOIE: Receiver timeout interrupt enable</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Interrupt inhibited</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.</td>
<td></td>
<td></td>
<td></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 33.4: USART implementation on page 1038.
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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

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 wake-up method

This bit determines the USART wake-up 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 is 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 33.4: USART implementation on page 1038.

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 CK pin is always available when CLKEN = 1, regardless of the UE bit value.

### 33.8.3  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>MSBFIRST</td>
<td>DATAINV</td>
<td>TXINV</td>
<td>RXINV</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>
<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>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>LBDIE</td>
<td>LBDL</td>
<td>ADDM7</td>
<td>DIS_NSS</td>
<td>Res.</td>
<td>Res.</td>
<td>SLVEN</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 31:24 $ADD[7:0]$: Address of the USART node
These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:
- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only $ADD[3:0]$ bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When $WUS[1:0]$ is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through $ADD[6:0]$ or $ADD[3:0]$ bitfield (depending on $ADDM7$ bit), and WUF interrupt is enabled by setting $WUFIE$ bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to $ADD[7:0]$ value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.
These bits can only be written when the reception is disabled (RE = 0) or when 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 33.4: USART implementation on page 1038.

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 and 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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

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 (VDD = 1/idle, Gnd = 0/mark)
1: TX pin signal values are inverted (VDD = 0/mark, Gnd = 1/idle).
This enables 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 (VDD = 1/idle, Gnd = 0/mark)
1: RX pin signal values are inverted (VDD = 0/mark, Gnd = 1/idle).
This enables 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 enables 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 33.4: USART implementation on page 1038.*

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 enables the user to enable the CK pin.
0: CK pin disabled
1: CK 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 33.4: USART implementation on page 1038.
In smartcard mode, in order to provide correctly the CK 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 enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship
0: Steady low value on CK pin outside transmission window
1: Steady high value on CK 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 33.4: USART implementation on page 1038.

Bit 9 **CPHA**: Clock phase
This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see Figure 326 and Figure 327)
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 33.4: USART implementation on page 1038.

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 CK pin in synchronous mode.
0: The clock pulse of the last data bit is not output to the CK pin
1: The clock pulse of the last data bit is output to the CK 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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.
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 33.4: USART implementation on page 1038.

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**: NSS pin enable
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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

*Note*: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

33.8.4  **USART control register 3 (USART_CR3)**
Address offset: 0x08
Reset value: 0x0000 0000
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
- 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
- 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 33.4: USART implementation on page 1038.

Bit 23 **TXFTIE:** TXFIFO threshold interrupt enable
- 0: Interrupt inhibited
- 1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE:** Wake-up from low-power mode interrupt enable
- 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 wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.
Bits 21:20  **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up 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).

*Note:* 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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.

Bit 14  **DEM**: Driver enable mode

This bit enables 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 33.4: USART implementation on page 1038.

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 is 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 is bypassed and data is 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 enables checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables 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 33.4: USART implementation on page 1038.

Bit 9 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS mode enabled, data is only transmitted when the RTS input is deasserted (tied to 0). If the RTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while RTS is asserted, the transmission is postponed until RTS is deasserted.

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 33.4: USART implementation on page 1038.

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 33.4: USART implementation on page 1038.*

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 33.4: USART implementation on page 1038.*

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 33.4: USART implementation on page 1038.*

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 33.4: USART implementation on page 1038.*

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 = 1 or 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.
### 33.8.5 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*

```

<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>

BRR[15:0]

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:

### 33.8.6 USART guard time and prescaler register (USART_GTPR)

*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>

GT[7:0] PSC[7:0]

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

Bits 15:0 **GT[7:0]:** Guard time

Bits 15:0 **PSC[7:0]:** Prescaler
### 33.8.7 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

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>
</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></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>
<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>
<td></td>
</tr>
</tbody>
</table>

#### 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 33.4: USART implementation on page 1038.

#### 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
- PSC[7:0] is used to program 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):

**In smartcard mode:**
- PSC[4:0] = Prescaler value
- PSC[4:0] is used to program 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 1 (IrDA mode) / by 2 (smartcard mode)
- 00010: Divides the source clock by 2 (IrDA mode) / by 4 (smartcard mode)
- 00011: Divides the source clock by 3 (IrDA mode) / by 6 (smartcard mode)
- ...
- 11111: Divides the source clock by 31 (IrDA mode) / by 62 (smartcard mode)
- 0010 0000: Divides the source clock by 32 (IrDA mode)
- ...
- 1111 1111: Divides the source clock by 255 (IrDA mode)

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 33.4: USART implementation on page 1038.
RM0434 Universal synchronous/asynchronous receiver transmitter (USART/UART)

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 EOBBCF 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 33.4: USART implementation on page 1038.

### 33.8.8 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>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>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>

STI

RM0434 Rev 13  1105/1530
33.8.9 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x00X80 00C0

X = 2 if FIFO/smartcard mode is enabled
X = 0 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>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td>Res.</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>EOF</td>
<td>RTOF</td>
<td>CTS</td>
<td>CTSIF</td>
<td>LBDF</td>
<td>TXFNF</td>
<td>TC</td>
<td>RXFNE</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>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

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 33.4: USART implementation on page 1038.
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 is 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 does 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 33.4: USART implementation on page 1038.

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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

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**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.*

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: Break character transmitted
1: Break character requested by setting SBKRQ bit in USART_RQR register

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 is also 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_RQR 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 33.4: USART implementation on page 1038.*

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_CR1 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 33.4: USART implementation on page 1038.*

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 is set.*

*Note: 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 CTS input pin.
0: CTS line set
1: CTS 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 CTS 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 CTS status line
1: A change occurred on the CTS 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 33.4: USART implementation on page 1038.

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 are 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 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 is immediately set.
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 is not 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 a software, writing 1 to the ORECF, in the USART_ICR register.
An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 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.
33.8.10 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>Address Offset</th>
<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>ACK</td>
<td>WUF</td>
<td>RWU</td>
<td>SBKF</td>
<td>CMF</td>
<td>BUSY</td>
<td>ACK</td>
<td>WUF</td>
<td>RWU</td>
<td>SBKF</td>
<td>CMF</td>
<td>BUSY</td>
<td>ACK</td>
<td>WUF</td>
<td>RWU</td>
<td>SBKF</td>
<td>CMF</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>ABRF</td>
<td>ABRE</td>
<td>UDR</td>
<td>EOBF</td>
<td>RTOF</td>
<td>CTS</td>
<td>TSBIF</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>
<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>
<td></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 33.4: USART implementation on page 1038.

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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.

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**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation on page 1038.
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: Break character transmitted
1: Break character requested by setting SBKRQ bit in USART_RQR register

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 (RXNE is also 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 ABRRQ bit 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 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 33.4: USART implementation on page 1038.*

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_CR1 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 33.4: USART implementation on page 1038.*
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 is 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 CTS input pin.
0: CTS line set
1: CTS 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 CTS 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 CTS status line
1: A change occurred on the CTS 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 33.4: USART implementation on page 1038.

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 is 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 is not set again until the RXNE bit has been set (i.e. a new idle line
occurs).
If mute mode is enabled (MM = 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 LPUART_CR1 register, or EIE = 1
in the USART_CR3 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.*
33.8.11 USART interrupt flag clear register (USART_ICR)

**Address offset: 0x20**

**Reset value: 0x0000 0000**

| Bit 31-21 | Reserved, must be kept at reset value. |
| Bit 20 WUCF | Wake-up 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 wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038. |

<p>| Bit 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. |</p>
<table>
<thead>
<tr>
<th>Bit 16:14</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 13</td>
<td><strong>UDRCF</strong>: SPI slave underrun clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the UDRF flag in the USART_ISR register.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038</td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>EOBCF</strong>: End of block clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the EOBF flag 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 33.4: USART implementation on page 1038.</td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>RTOCF</strong>: Receiver timeout clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the RTOF flag in the USART_ISR register.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.</td>
</tr>
<tr>
<td>Bit 10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9</td>
<td><strong>CTSCF</strong>: CTS clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>LBDCF</strong>: LIN break detection clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the LBDF flag in the USART_ISR register.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.</td>
</tr>
<tr>
<td>Bit 7</td>
<td><strong>TCBGTCF</strong>: Transmission complete before Guard time clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.</td>
</tr>
<tr>
<td>Bit 6</td>
<td><strong>TCCF</strong>: Transmission complete clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the TC flag in the USART_ISR register.</td>
</tr>
<tr>
<td>Bit 5</td>
<td><strong>TXFECF</strong>: TXFIFO empty clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the TXFE flag in the USART_ISR register.</td>
</tr>
<tr>
<td>Bit 4</td>
<td><strong>IDLECF</strong>: Idle line detected clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the IDLE flag in the USART_ISR register.</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>ORECF</strong>: Overrun error clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the ORE flag in the USART_ISR register.</td>
</tr>
<tr>
<td>Bit 2</td>
<td><strong>NECF</strong>: Noise detected clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the NE flag in the USART_ISR register.</td>
</tr>
<tr>
<td>Bit 1</td>
<td><strong>FECF</strong>: Framing error clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the FE flag in the USART_ISR register.</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>PECF</strong>: Parity error clear flag</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the PE flag in the USART_ISR register.</td>
</tr>
</tbody>
</table>
33.8.12 USART receive data register (USART_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>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 31:9 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 320).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

33.8.13 USART transmit data register (USART_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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 8:0 **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 320).
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/ TXFN F = 1.
33.8.14 USART prescaler register (USART_PRESC)

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>
</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 31:4 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 is 1011 i.e. input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 1038.
## 33.8.15 USART register map

The table below gives the USART register map and reset values.

### Table 212. USART register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>USART_RTOR</td>
<td>RES</td>
<td>RTO[23:0]</td>
</tr>
<tr>
<td>0x00</td>
<td>USART_RQR</td>
<td>RES</td>
<td>RQR</td>
</tr>
<tr>
<td>0x08</td>
<td>USART_RICR</td>
<td>RES</td>
<td>RICR</td>
</tr>
<tr>
<td>0x10</td>
<td>USART_BRR</td>
<td>RES</td>
<td>BRR[15:0]</td>
</tr>
<tr>
<td>0x1C</td>
<td>USART_RICR</td>
<td>RES</td>
<td>RICR</td>
</tr>
<tr>
<td>0x20</td>
<td>USART_IER</td>
<td>RES</td>
<td>IER</td>
</tr>
<tr>
<td>0x20</td>
<td>USART_ICR</td>
<td>RES</td>
<td>ICR</td>
</tr>
<tr>
<td>0x20</td>
<td>USART_IDR</td>
<td>RES</td>
<td>IDR</td>
</tr>
<tr>
<td>0x2C</td>
<td>USART_IER</td>
<td>RES</td>
<td>IER</td>
</tr>
<tr>
<td>0x2D</td>
<td>USART_IDR</td>
<td>RES</td>
<td>IDR</td>
</tr>
<tr>
<td>0x30</td>
<td>USART_SR</td>
<td>RES</td>
<td>SR</td>
</tr>
<tr>
<td>0x34</td>
<td>USART_SR</td>
<td>RES</td>
<td>SR</td>
</tr>
</tbody>
</table>

### Description

- **Offset**: The offset value of the register in memory.
- **Register name**: The name of the register.
- **Reset value**: The reset value of the register.

### Notes

- The reset values are given in hexadecimal format.
- The table includes registers for various control and status functions, such as receive FIFO enable, transmit FIFO enable, data format, etc.
- Detailed descriptions of each register are provided in the document.
Refer to *Section 2.2: Memory organization* for the register boundary addresses.
34 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

34.1 LPUART introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the device 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.
34.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
  - From 300 baud to 9600 baud 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: wake-up from mute mode by idle line detection or address mark detection
34.3 LPUART implementation

The table(s) below describe(s) LPUART implementation. It(they) also include(s) USARTs for comparison.

**Table 213. USART / LPUART features**

<table>
<thead>
<tr>
<th>USART / LPUART modes/features(1)</th>
<th>USART1</th>
<th>LPUART1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous mode (Master/Slave)</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire half-duplex communication</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain and wake-up from low-power mode</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>USART data length</td>
<td>7, 8 and 9 bits</td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

1. X = supported.
34.4 LPUART functional description

34.4.1 LPUART block diagram

The simplified block diagram given in Figure 347 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.
34.4.2 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.**

Refer to **Table 214** and **Table 215** for the list of USART input/output pins and internal signals.

### Table 214. LPUART input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPUART_RX</td>
<td>Input</td>
<td>Serial data receive input</td>
</tr>
<tr>
<td>LPUART_TX</td>
<td>Output</td>
<td>Transmit data output</td>
</tr>
<tr>
<td>LPUART_CTS</td>
<td>Input</td>
<td>Clear to send</td>
</tr>
<tr>
<td>LPUART_RTS</td>
<td>Output</td>
<td>Request to send</td>
</tr>
<tr>
<td>LPUART_DE(1)</td>
<td>Output</td>
<td>Driver enable</td>
</tr>
</tbody>
</table>

1. LPUART_DE and LPUART_RTS share the same pin.

### Description of USART input/output signals

#### Table 215. LPUART internal input/output signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpuart_pclk</td>
<td>Input</td>
<td>APB clock</td>
</tr>
<tr>
<td>lpuart_ker_ck</td>
<td>Input</td>
<td>USART kernel clock</td>
</tr>
</tbody>
</table>
34.4.3 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 321).

- 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 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 clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.
34.4.4 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 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 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 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 LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag is 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 does 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 bits fields.
  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

34.4.5 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 347).

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 corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.
An idle frame is 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 is supported by normal LPUART, single-wire and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 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 349. 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 34.4.12: Continuous communication using DMA and LPUART.
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 device to enter the low-power mode (see *Figure 350: TC/TXE behavior when transmitting*).
Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see Figure 348).

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.

34.4.6 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.

Figure 350. TC/TXE behavior when transmitting

Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.
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 34.4.12: Continuous communication using DMA and LPUART.
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 LPUART 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 is not lost. The previous data is available when a read to LPUART\_RDR is performed;
- the shift register is 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 is not lost. It is available when a read to LPUART\_RDR is performed.
- the shift register is 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. 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 wake-up 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_PRESC register.

Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up 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 enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

*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 is 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.

### 34.4.7 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>
<thead>
<tr>
<th>Table 216. Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32.768 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud rate</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>S.No</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
34.4.8 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)

\[
\text{LPUART receiver tolerance} = \text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU}
\]

where

- **DWU** is the error due to sampling point deviation when the wake-up from low-power mode is used.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 218:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.
Table 218. Tolerance of the LPUART receiver

<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>1.69%</td>
<td>2.33%</td>
<td>2.53%</td>
<td>4.14%</td>
</tr>
<tr>
<td>7 bits (M = 01'), 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 218 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').

### 34.4.9 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 352.

![Figure 352. Mute mode using Idle line detection](MSv31154V1)

**Note:** If the MMRQ is set while the IDLE character has already elapsed, the mute mode is not 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-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 353.

**Figure 353. Mute mode using address mark detection**

![Figure 353](MSv31888V1)

**34.4.10 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 219.

**Table 219: LPUART frame formats**

<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>SB</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>SB</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>SB</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>SB</td>
</tr>
</tbody>
</table>

2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

**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 equal 00110101, and 4 bits are set, then the parity bit is equal to 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 equal 00110101 and 4 bits set, then the parity bit is equal to 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)).

34.4.11 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.

34.4.12 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 33.4: USART implementation on page 1038 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 33.5.6. 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 354. Transmission using DMA**

*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.

![Figure 355. Reception using DMA](image)

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.

### 34.4.13 RS232 hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The Figure 342 shows how to connect 2 devices in this mode:

**Figure 356. Hardware flow control between 2 LPUARTs**

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 RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. Figure 357 shows an example of communication with RTS flow control enabled.

**Figure 357. RS232 RTS flow control**

**Note:** When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.
RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (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 CTS is asserted 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 CTS 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 358 shows an example of communication with CTS flow control enabled.

**Figure 358. RS232 CTS flow control**

<table>
<thead>
<tr>
<th>CTS</th>
<th>Transmit data register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transmit data register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 1 Start bit Data 2 Stop bit Data 3 Stop bit</td>
</tr>
</tbody>
</table>

Writing data 3 in TDR

Transmission of Data 3 is delayed until CTS = 0

Note: For correct behavior, CTS must be deasserted 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 enables 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 deassertion 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 deassertion 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 \(P = \text{BRR}[20:11]\)

### 34.4.14 LPUART low-power management

The LPUART has advanced low-power mode functions that enable 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 luart_pclk is gated, the LPUART provides a wake-up interrupt (luart_wkup) if a specific action requiring the activation of the luart_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 luart_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 luart_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 enables 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 luart_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 wake-up 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 enables doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.
Alternatively, a specific luart_wkup interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and luart_wkup interrupt is generated if the WUFIE bit is set.

**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 wake-up 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 wake-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 LPUART is put into mute mode before entering low-power mode:

Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.

If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

**Note:** When FIFO management is enabled, mute mode is used with wake-up 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).

**Wake-up from low-power mode when LPUART kernel clock luart_ker_ck is OFF in low-power mode**

If during low-power mode, the luart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the luart_ker_ck clock to be switched ON thanks to the luart_ker_ck_req signal. The luart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the luart_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 wake-up event programmed to "address match detection" and FIFO management disabled.

*Figure 359* shows the behavior when the wake-up event is verified.
Figure 359. Wake-up event verified (wake-up event = address match, FIFO disabled)

Address match event WUF = '1'
LPUART sends a wakeup event to the MCU

Data reception goes on

Figure 360 shows the behavior when the wake-up event is not verified.

Figure 360. Wake-up event not verified (wake-up event = address match, FIFO disabled)

Address does not match

Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.
Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see Section 34.4.8: 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 218: Tolerance of the LPUART receiver, the LPUART receiver tolerance equals 3.41%.

\[ D_{WU_{\text{max}}} = t_{WULPUART}/ (11 \times T_{\text{bit Min}}) \]

\[ T_{\text{bit Min}} = t_{WULPUART}/ (11 \times D_{WU_{\text{max}}}) \]

where \( t_{WULPUART} \) is the wake-up 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 \( \text{lpuart\_ker\_ck} \) inaccuracy.

For example, if HSI is used as \( \text{lpuart\_ker\_ck} \), 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).

\[ D_{WU_{\text{max}}} = 3.41\% - 1\% = 2.41\% \]

\[ T_{\text{bit min}} = 3 \mu s/ (11 \times 2.41\%) = 11.32 \mu s. \]

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: \( 1/11.32 \mu s = 88.36 \text{kbaud} \).

34.5 LPUART in low-power modes

Table 220. Effect of low-power modes on the LPUART

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. LPUART interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^{(1)})</td>
<td>The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 34.3: LPUART implementation to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.
# LPUART interrupts

Refer to Table 221 for a detailed description of all LPUART interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop(1) modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>Write TDR</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit FIFO Not Full</td>
<td>TXFN</td>
<td>TXFNIE</td>
<td>TXFIFO full</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit FIFO Empty</td>
<td>TXF</td>
<td>TXFEIE</td>
<td>Write TDR or write 1 in TXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>Write TDR</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>Write 1 in CTSCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission Complete</td>
<td>TC</td>
<td>TIE</td>
<td>Write TDR or write 1 in TCCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>Read RDR or write 1 in RXFRQ</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>Read RDR until RXFIFO empty or write 1 in RXFRQ</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive FIFO Full</td>
<td>RXFF(2)</td>
<td>RXFFIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RXNEIE/RXFNEIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>Write 1 in IDLECF</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>Write 1 in PECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noise error in multibuffer communication.</td>
<td>NE</td>
<td>Write 1 in NFCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overrun error in multibuffer communication.</td>
<td>ORE(3)</td>
<td>EIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framing Error in multibuffer communication.</td>
<td>FE</td>
<td>Write 1 in FECF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>Write 1 in CMCF</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wake-up from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>Write 1 in WUC</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to Section 34.3: LPUART implementation for the list of supported Stop modes.

2. 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 is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

3. When OVRDIS = 0.

34.7 LPUART registers

Refer to Section 1.2 on page 61 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

34.7.1 LPUART control register 1 (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</th>
<th>RXFFIE: RXFIFO full interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Interrupt is inhibited</td>
</tr>
<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</th>
<th>TXFEIE: TXFIFO empty interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Interrupt is inhibited</td>
</tr>
<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</th>
<th>FIFOEN: 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>

The peripheral registers have to be accessed by words (32 bits).
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': 1 Start bit, 8 Data bits, n Stop bit
- M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit
- M[1:0] = '10': 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 33.5.20: 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 34.4.13: 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 wake-up method
This bit determines the LPUART wake-up 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 is 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.

### 34.7.2 LPUART 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>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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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>Res.</td>
<td>CMIE</td>
<td>MME</td>
<td>MD</td>
<td>WAKE</td>
<td>PCE</td>
<td>PS</td>
<td>PEIE</td>
<td>TXEIE</td>
<td>TCIE</td>
<td>RXNEIE</td>
<td>IDLEIE</td>
<td>TE</td>
<td>RE</td>
<td>UESM</td>
<td>UE</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 31:30  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': 1 Start bit, 8 Data bits, n Stop bit
- M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit
- M[1:0] = '10': 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 to Section 33.5.20: 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 to Section 34.4.13: 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 wake-up method
This bit determines the LPUART wake-up 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 is 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: An 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
34.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 3 TE</th>
<th>Transmitter enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit enables the transmitter. It is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>0: Transmitter is disabled</td>
<td></td>
</tr>
<tr>
<td>1: Transmitter is enabled</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> 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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2 RE</th>
<th>Receiver enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit enables the receiver. It is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>0: Receiver is disabled</td>
<td></td>
</tr>
<tr>
<td>1: Receiver is enabled and begins searching for a start bit</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1 UESM</th>
<th>LPUART enable in Stop mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>This bit is set and cleared by software.</td>
<td></td>
</tr>
<tr>
<td>0: LPUART not able to wake up the MCU from low-power mode.</td>
<td></td>
</tr>
<tr>
<td>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)</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0 UE</th>
<th>LPUART enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>0: LPUART prescaler and outputs disabled, low-power mode</td>
<td></td>
</tr>
<tr>
<td>1: LPUART enabled</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> 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.</td>
<td></td>
</tr>
<tr>
<td>The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:24 **ADD[7:0]:** Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol); the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set. These bits can only be written when the reception is disabled (RE = 0) or when the USART 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 ($V_{DD} = 1$/idle, $Gnd = 0$/mark)
1: TX pin signal values are inverted ($V_{DD} = 0$/mark, $Gnd = 1$/idle).

This enables 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 ($V_{DD} = 1$/idle, $Gnd = 0$/mark)
1: RX pin signal values are inverted ($V_{DD} = 0$/mark, $Gnd = 1$/idle).

This enables 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 enables 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.**
34.7.4 **LPUART 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>OVRDI</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></td>
<td></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.
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**: Wake-up 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 wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation.*

Bits 21:20  **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up 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 wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 33.4: 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 enables 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 is 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 enables 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 CTS input is deasserted (tied to 0).
If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
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 RTS output is deasserted (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.

### 34.7.5 LPUART 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>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>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 31:20 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 \( \geq 0x300 \) 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 \([3 \times \text{baud rate}..4096 \times \text{baud rate}]\).*
34.7.6 LPUART request register (LPUART_RQR)

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</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>w</td>
<td>w</td>
<td>w</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 31:5 Reserved, must be kept at reset value.

- **Bit 4 TXFRQ**: Transmit data flush request
  
  This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

  **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 clears the RXNE flag. This enables discarding the received data without reading it, and avoid an overrun condition.

- **Bit 2 MMRQ**: Mute mode request
  
  Writing 1 to this bit puts the LPUART 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**: 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.

- **Bit 0** Reserved, must be kept at reset value.

34.7.7 LPUART interrupt and status register (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>Bits</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>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>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 31:28  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 wake-up 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**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up 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 WUFIN = 1 in the LPUART_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value*
Bit 19  **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up 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 wake-up 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: Break character transmitted
1: Break character requested by setting SBKRQ bit in LPUART_RQR register

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 CTS input pin.

0: CTS line set
1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9  **CTSF**: CTS interrupt flag

This bit is set by hardware when the CTS 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 CTS status line
1: A change occurred on the CTS 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**: TX FIFO not full

TXFNF is set by hardware when TX FIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TX FIFO. This flag remains set until the TX FIFO is full. When the TX FIFO 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 TX FIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TX FIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFE 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 TCE = 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 is set immediately.

Bit 5 **RXFNE**: RX FIFO not empty

RXFNE bit is set by hardware when the RX FIFO 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 RX FIFO. It is cleared when the RX FIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ 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 is not 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.*

---

**Low-power universal asynchronous receiver transmitter (LPUART)**

---

**RM0434 Rev 13 1167/1530**
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, or EIE = 1 in the LPUART_CR3 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_CR3 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.*
34.7.8 LPUART 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>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>Res</td>
<td>CTS</td>
<td>CTSIF</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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:23 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 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 wake-up 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**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up 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 wake-up 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: Break character transmitted
1: Break character requested by setting SBKRQ bit in LPUART_RQR register

Bit 17 **CMF**: Character match flag
This bit is set by hardware, when 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 CTS input pin.
0: CTS line set
1: CTS 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 CTS 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 CTS status line
1: A change occurred on the CTS 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 is immediately set.

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 is not 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 a 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, or EIE = 1 in the LPUART_CR3 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_CR3 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

### 34.7.9 LPUART interrupt flag clear register (LPUART_ICR)

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>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 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

*Note:* if the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation.

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 LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.
34.7.10 LPUART receive data register (LPUART_RDR)

Address offset: 0x24
Reset value: 0x0000 0000

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC 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 IDLE 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.

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 347).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

34.7.11 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28
Reset value: 0x0000 0000

Bits 15:0 **TDR[15:0]**: Transmit data value
Contains the transmitted data character.
34.7.12 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

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>PRESCALER[3:0]</td>
</tr>
<tr>
<td>rw rw rw rw</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 TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 347).
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.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.
If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 34.3: LPUART implementation on page 1125.
## LPUART register map

The table below gives the LPUART register map and reset values.

**Table 222. LPUART register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register map</th>
<th>Reset value</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>LPUART_CR1</td>
<td>FIFO mode enabled</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td>LPUART_CR1</td>
<td>FIFO mode disabled</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>LPUART_CR2</td>
<td>ADD[7:0]</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>LPUART_CR3</td>
<td>TXFCF[52:0] RXFCF[52:0] WUS[1:0]</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>LPUART_BRR</td>
<td>BRR[19:0]</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x10- 0x14</td>
<td>LPUART_RQR</td>
<td>Reserved</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>LPUART_ISR</td>
<td>TXF RXF REACK TXF TEACK WUF RXU SIKF CME BUSY</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>LPUART_ISR</td>
<td>FIFO mode enabled</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>LPUART_ISR</td>
<td>FIFO mode disabled</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>LPUART_ICR</td>
<td>Reserved</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>LPUART_RDR</td>
<td>RDR[8:0]</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>LPUART_TDR</td>
<td>TDR[8:0]</td>
<td>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table provides a detailed view of the LPUART register map, including the reset values for each register offset. The registers are categorized into groups such as FIFO mode enabled and FIFO mode disabled, along with additional specific function registers like add(7:0) and ADD[7:0]. The reset values are provided for each register, ensuring a clear understanding of the initial state of each register after a reset.
Refer to Section 2.2: Memory organization for the register boundary addresses.
35 Serial peripheral interface (SPI)

35.1 Introduction

The SPI interface can be used to communicate with external devices using the SPI protocol. SPI 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.

35.2 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
35.3 SPI implementation

The following table describes all the SPI instances and their features embedded in the devices.

<table>
<thead>
<tr>
<th>SPI Features</th>
<th>SPI1</th>
<th>SPI2(1)</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>No</td>
<td>No</td>
</tr>
<tr>
<td>Hardware CRC calculation</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data size configurable</td>
<td>from 4 to 16 bits</td>
<td>from 4 to 16 bits</td>
</tr>
<tr>
<td>Rx/Tx FIFO size</td>
<td>32 bits</td>
<td>32 bits</td>
</tr>
<tr>
<td>Wake-up capability from Low-power Sleep</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. Available only on STM32WB55xx devices.

35.4 SPI functional description

35.4.1 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 361.

Figure 361. 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 35.4.5: 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.

### 35.4.2 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 362. Full-duplex single master/ single slave application](image-url)

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 35.4.5: 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.

Figure 363. Half-duplex single master/ single slave application

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 35.4.5: Slave select (NSS) pin management.

2. In this configuration, the master’s MISO pin and the slave’s MOSI pin can be used as GPIOs.

3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectional mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

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_CR1 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 35.4.5: 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.
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 35.4.5: 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. OVR 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).

35.4.3 Standard multislave 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 365). 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.
35.4.4 Multimaster communication

Unless SPI bus is not designed for a multimaster 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 366. Multimaster 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.

### 35.4.5 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.
35.4.6 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 368, 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 369). During communication, only bits within the data frame are clocked and transferred.
35.4.7 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 cannot 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 CPHA 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.

35.4.8 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 & BIDIOE = 0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated section.

35.4.9 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 35.4.14: 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 35.4.13: 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, RXFIFO can store up to four data frames, whereas 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 371 through Figure 374.
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 35.4.10: 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 (BIDMODE=1, BIDIOE=0) or simplex (BIDMODE=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 35.4.5: 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 370 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 370. 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 TXDMAEN or RXDMAEN 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 371 through Figure 374**.

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 1189.)
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 371 on page 1193 through Figure 374 on page 1196:

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 ⅔ 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 1192 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 1192 for details about common assumptions and notes.
Figure 373. Master full-duplex communication with CRC

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 1192 for details about common assumptions and notes.
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 1192 for details about common assumptions and notes.
35.4.10 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.
35.4.11 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 35.4.14: 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.

### 35.4.12 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 375 illustrates NSS pin management when NSSP pulse mode is enabled.

**Figure 375. NSSP pulse generation in Motorola SPI master mode**

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.

### 35.4.13 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 376). 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

![Image of NSSP pulse generation in Motorola SPI master mode]

**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.
baud rate value set in through the BR[2:0] bits in the SPIx_CR1 register. It is given by the formula:

\[
\frac{t_{\text{baud rate}}}{2} + 4 \cdot t_{pclk} + t_{\text{release}} \cdot \frac{t_{\text{baud rate}}}{2} + 6 \cdot t_{pclk}
\]

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 376: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

---

**35.4.14 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 follows 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.

### 35.5 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 224. SPI interrupt requests

<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></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>
35.6 SPI 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.

35.6.1 SPI control register 1 (SPIx_CR1)

Address offset: 0x00
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>BIDIMODE:</td>
</tr>
<tr>
<td>14</td>
<td>BIDIOE:</td>
</tr>
<tr>
<td>13</td>
<td>CRCEN:</td>
</tr>
<tr>
<td>12</td>
<td>CRCNEXT:</td>
</tr>
<tr>
<td>11</td>
<td>CRCL:</td>
</tr>
</tbody>
</table>

Bit 15 **BIDIMODE**: Bidirectional data mode enable.
- This bit enables half-duplex communication using common single bidirectional data line.
- Keep RXONLY bit clear when bidirectional mode is active.
  - 0: 2-line unidirectional data mode selected
  - 1: 1-line bidirectional data mode selected

Bit 14 **BIDIOE**: Output enable in bidirectional mode
- This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode.
  - 0: Output disabled (receive-only mode)
  - 1: Output enabled (transmit-only mode)
- Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.

Bit 13 **CRCEN**: Hardware CRC calculation enable
- 0: CRC calculation disabled
- 1: CRC calculation enabled
- Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.

Bit 12 **CRCNEXT**: Transmit CRC next
- 0: Next transmit value is from Tx buffer.
- 1: Next transmit value is from Tx CRC register.
- Note: This bit has to be written as soon as the last data is written in the SPIx_DR register.

Bit 11 **CRCL**: CRC length
- This bit is set and cleared by software to select the CRC length.
  - 0: 8-bit CRC length
  - 1: 16-bit CRC length
- Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.
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 multi-slave 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)

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 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 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 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 1188.

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.

Bit 2  **MSTR:** Master selection

    0: Slave configuration
    1: Master configuration

    *Note:* This bit should not be changed when communication is ongoing.
35.6.2  SPI control register 2 (SPIx_CR2)

Address offset: 0x04
Reset value: 0x0700

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved</td>
<td>Unused</td>
</tr>
<tr>
<td>14</td>
<td>LDMA_TX</td>
<td>Last DMA transfer for transmission</td>
</tr>
<tr>
<td>13</td>
<td>LDMA_RX</td>
<td>Last DMA transfer for reception</td>
</tr>
<tr>
<td>12</td>
<td>FRXTH</td>
<td>FIFO reception threshold</td>
</tr>
<tr>
<td>11</td>
<td>DS[3:0]</td>
<td>Data shift register</td>
</tr>
<tr>
<td>10</td>
<td>TXIEI</td>
<td>Transmit empty interrupt enable</td>
</tr>
<tr>
<td>9</td>
<td>RXNEIE</td>
<td>Receive empty interrupt enable</td>
</tr>
<tr>
<td>8</td>
<td>ERRIE</td>
<td>Error interrupt enable</td>
</tr>
<tr>
<td>7</td>
<td>FRF</td>
<td>Function register</td>
</tr>
<tr>
<td>6</td>
<td>NSSP</td>
<td>NSS pulse</td>
</tr>
<tr>
<td>5</td>
<td>SSOE</td>
<td>Slave selection active</td>
</tr>
<tr>
<td>4</td>
<td>TXDMAEN</td>
<td>Transmitter DMA enable</td>
</tr>
<tr>
<td>3</td>
<td>RXDMAEN</td>
<td>Receiver DMA enable</td>
</tr>
</tbody>
</table>

Bit 15  Reserved, must be kept at reset value.

Bit 14  **LDMA_TX**: 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 TXDMAEN 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 1188 if the CRCEN bit is set.*

Bit 13  **LDMA_RX**: 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 1188 if the CRCEN bit is set.*

Bit 12  **FRXTH**: 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)
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)

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 **RXNIEIE**: 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).

- 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).*

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 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 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

### 35.6.3 SPI status register (SPIx_SR)

Address offset: 0x08
Reset value: 0x0002

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Res.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>FTLVL[1:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>FRLVL[1:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>FRE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>BSY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>OVR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>MODF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CRCE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RR</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>TXE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>RXNE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Res.</td>
<td></td>
<td></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)

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 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. Refer to *Section 35.4.11: SPI 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 not busy  
- 1: SPI 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 35.4.10: SPI status flags and Procedure for disabling the SPI on page 1188.

Bit 6 **OVR**: Overrun flag  
- 0: No overrun occurred  
- 1: Overrun occurred  
This flag is set by hardware and reset by a 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 1198 for the software sequence.

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.

Bits 3:2 Reserved, must be kept at reset value.

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

### 35.6.4 SPI data register (SPIx_DR)

Address offset: 0x0C  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>DR[15:0]: Data register</td>
</tr>
</tbody>
</table>
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 35.4.9: 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.  

---

1208/1530  
RM0434 Rev 13
35.6.5 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.

35.6.6 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>

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.

A read to this register when the BSY Flag is set could return an incorrect value.

35.6.7 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.

> A read to this register when the BSY flag is set could return an incorrect value.
35.6.8 SPI register map

Table 225 shows the SPI register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</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>0x00</td>
<td>SPIx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>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></td>
</tr>
<tr>
<td>0x04</td>
<td>SPIx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</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>
</tr>
<tr>
<td>0x08</td>
<td>SPIx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>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>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>SPIx_DR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>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></td>
</tr>
<tr>
<td>0x10</td>
<td>SPIx_CRCPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>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></td>
</tr>
<tr>
<td>0x14</td>
<td>SPIx_RXCRCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>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></td>
</tr>
<tr>
<td>0x18</td>
<td>SPIx_TXCRCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>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></td>
</tr>
</tbody>
</table>

Refer to Section 2.2 on page 66 for the register boundary addresses.
36  Serial audio interface (SAI)

36.1  Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many stereo or mono audio applications may be targeted. I2S standards, LSB or MSB-justified, PCM/DSP, TDM, and AC’97 protocols may be addressed for example. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has its own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

36.2  SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO.
- 8-word integrated FIFOs for each audio subblock.
- Synchronous or asynchronous mode between the audio subblocks.
- Master or slave configuration independent for both audio subblocks.
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode.
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit.
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required.
- Up to 16 slots available with configurable size.
- Number of bits by frame can be configurable.
- Frame synchronization active level configurable (offset, bit length, level).
- First active bit position in the slot is configurable.
- LSB first or MSB first for data transfer.
- Mute mode.
- Stereo/Mono audio frame capability.
- Communication clock strobing edge configurable (SCK).
- Error flags with associated interrupts if enabled respectively.
  - Overrun and underrun detection,
  - Anticipated frame synchronization signal detection in slave mode,
  - Late frame synchronization signal detection in slave mode,
  - Codec not ready for the AC’97 mode in reception.
- Interrupt sources when enabled:
  - Errors,
  - FIFO requests.
2-channel DMA interface.

36.3 SAI implementation

Table 226. STM32WB55xx SAI features

<table>
<thead>
<tr>
<th>SAI features</th>
<th>SAI1</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97</td>
<td>X</td>
</tr>
<tr>
<td>FIFO size</td>
<td>8 words</td>
</tr>
<tr>
<td>SPDIF</td>
<td>X</td>
</tr>
<tr>
<td>PDM</td>
<td>X(2)(3)</td>
</tr>
</tbody>
</table>

1. 'X' = supported, '-' = not supported.
2. Only signals D[3:1], and CK[2:1] are available.
3. The DLYM4R[2:0] and DLYM4L[2:0] bitfields shall not be used and MICNBR[1:0] shall not be set to 0b11.

36.4 SAI functional description

36.4.1 SAI block diagram

Figure 377 shows the SAI block diagram while Table 227 and Table 228 list SAI internal and external signals.
The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC’97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).
36.4.2 SAI pins and internal signals

Table 227. SAI internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sai_a_gbl_it/</td>
<td>Output</td>
<td>Audio block A and B global interrupts.</td>
</tr>
<tr>
<td>sai_b_gbl_it</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sai_a_dma,</td>
<td>Input/output</td>
<td>Audio block A and B DMA acknowledges and requests.</td>
</tr>
<tr>
<td>sai_b_dma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sai_a_ker_ck/</td>
<td>Input</td>
<td>Audio block A/B kernel clock.</td>
</tr>
<tr>
<td>sai_b_ker_ck</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sai_pclk</td>
<td>Input</td>
<td>APB clock.</td>
</tr>
</tbody>
</table>

Table 228. SAI input/output pins

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAI_SCK_A/B</td>
<td>Input/output</td>
<td>Audio block A/B bit clock.</td>
</tr>
<tr>
<td>SAI_MCLK_A/B</td>
<td>Output</td>
<td>Audio block A/B master clock.</td>
</tr>
<tr>
<td>SAI_SD_A/B</td>
<td>Input/output</td>
<td>Data line for block A/B.</td>
</tr>
<tr>
<td>SAI_FS_A/B</td>
<td>Input/output</td>
<td>Frame synchronization line for audio block A/B.</td>
</tr>
<tr>
<td>SAI_D[3:1]</td>
<td>Input</td>
<td>PDM bitstream data(1).</td>
</tr>
</tbody>
</table>

1. These signals might not be available in all SAI instances. Refer to Section 36.3: SAI implementation for details.

36.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.
Slave mode
The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.
It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes
Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:
1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold (FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

36.4.4 SAI synchronization mode
SAI subclock A and B can be synchronized.

Internal synchronization
An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).
Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

36.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

36.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. Figure 378 illustrates this flexibility.

Figure 378. Audio frame

In AC’97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC’97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

Master mode

The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z.
 depending on the state of bit TRIS in the SAI_xCR2 register (refer to FS signal role). In
reception mode, the remaining bit is ignored.
If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure
that an audio frame contains an integer number of MCLK pulses per bit clock cycle.
If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to
Section 36.4.8: SAI clock generator
Slave mode
The audio frame length is mainly used to specify to the slave the number of bit clock
cycles per audio frame sent by the external master. It is used mainly to detect from the
master any anticipated or late occurrence of the frame synchronization signal during an
ongoing audio frame. In this case, an error is generated. For more details, refer to
Section 36.4.14: Error flags.
In slave mode, there are no constraints on the FRL[7:0] configuration in the
SAI_xFRCR register.
The number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity
The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a
frame is started. The start of the frame is edge sensitive.
In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The
start of the frame is synchronized to this signal. It is effective only if the start of the frame is
not detected during an ongoing communication and assimilated to an anticipated start of
frame (refer to Section 36.4.14: Error flags).
In master mode, the frame synchronization is sent continuously each time an audio frame is
complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in
the FIFO at the end of the previous audio frame, an underrun condition is managed as
described in Section 36.4.14: Error flags, but the audio communication flow is not
interrupted.

Frame synchronization active level length
The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the
active level of the frame synchronization signal. The length can be set from 1- to 128-bit
clock cycles.
As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified
modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset
Depending on the audio protocol targeted in the application, the frame synchronization
signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is
the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF
bit in the SAI_xFRCR register enables the possibility to choose between two configurations.
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

0: start of frame, like, for instance, the PCM/DSP, TDM, AC’97, audio protocols,

1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 379. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

1. The frame length must be even.

If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:
If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,

If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

**Figure 380. FS role is start of frame (FSDEF = 0)**

The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

### 36.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

For AC’97 protocol or SPDIF (when bit PRTC[1:0] = 10 or PRTC[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to Output data line management on an inactive slot) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

The slot size is also configurable as shown in Figure 381. The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOT register. The size is applied identically for each slot in an audio frame.
It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤ (SLOTSZ - DS),
- DS ≤ SLOTSZ,
- NBSLOT \times SLOTSZ ≤ FRL (frame length),

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC’97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in Section 36.4.11: AC’97 link controller.
### 36.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

**Generation of the master clock (MCLK_x)**

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

*Table 229* shows MCLK_x activation conditions.

<table>
<thead>
<tr>
<th>MCLKEN</th>
<th>NODIV</th>
<th>SAIEN for block x</th>
<th>MCLK_x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td></td>
<td>Enabled</td>
</tr>
</tbody>
</table>

*Note:* MCLK_x can also be generated in AC’97 mode, when MCLKEN is set to 1.
Generation of the bit clock (SCK_x)

The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

**Figure 383** illustrates the architecture of the audio block clock generator.

The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.
Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

\[ F_{MCLK_x} = \begin{cases} 256 \times F_{FS_x} & \text{if } OSR = 0 \\ 512 \times F_{FS_x} & \text{if } OSR = 1 \end{cases} \]

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

\[ F_{MCLK_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV} \]

The frame synchronization frequency is given by:

\[ F_{FS_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV \times OSR + 1} \times 256 \]

The bit clock frequency (SCK_x) is given by the following formula:

\[ F_{SCK_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV \times OSR + 1} \times \frac{FRL + 1}{256} \]

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^8.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

\[ F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x \_ker \_ck}}{MCKDIV} \]

The frequency of the frame synchronization (FS_x) in given by the following formula:

\[ F_{FS_x} = \frac{F_{sai_x \_ker \_ck}}{FRL + 1} \times \frac{MCKDIV}{256} \]

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.
Clock generator programming examples

*Table 230* gives programming examples for 48, 96 and 192 kHz.

### Table 230. Clock generator programming examples

<table>
<thead>
<tr>
<th>Input sa1_x_ker_ck clock frequency</th>
<th>MCLK</th>
<th>F_{MCLK}/F_{FS}</th>
<th>FRL (1)</th>
<th>OSR</th>
<th>NODIV</th>
<th>MCKEN</th>
<th>MCKDIV[5:0]</th>
<th>Audio Sampling frequency (F_{FS})</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.304 MHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>512</td>
<td>2^{N-1}</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 or 1</td>
<td></td>
<td>192 kHz</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2^{N-1}</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td>96 kHz</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>2^{N-1}</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td></td>
<td>48 kHz</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>2^{N-1}</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
<td>192 kHz</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>2^{N-1}</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td></td>
<td>96 kHz</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>2^{N-1}</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td></td>
<td>48 kHz</td>
</tr>
<tr>
<td>N</td>
<td>63</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td></td>
<td>192 kHz</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td></td>
<td>96 kHz</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td></td>
<td>48 kHz</td>
</tr>
</tbody>
</table>

1. N is an integer value between 3 and 8.

### 36.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

### Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).
When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

**Interrupt generation in reception mode**

The interrupt generation depends on the FIFO configuration in reception mode:

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).
interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xSR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

36.4.10 PDM interface

The PDM (Pulse Density Modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to Section 36.3: SAI implementation).

Figure 384 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 384. PDM typical connection and timing

1. \( n \) refers to the number of data lines and \( p \) to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the TDM interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).
The data processing sequence into the PDM is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the TDM interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A via the serial data line of the TDM interface.

*Figure 385* below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

**Note:** The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

1. *n* refers to the number of data lines and *p* to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to *Section 36.3: SAI implementation* for details.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.
To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

**Enabling the PDM interface**

To enable the PDM interface, follow the sequence below:

1. Configure SAI_A in TDM master mode (see Table 231).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

**Note:** Once the PDM interface and SAI_A are enabled, the first two TDMA frames received on SAI_ADR are invalid and must be dropped.

**Startup sequence**

*Figure 386* shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.
**SAI_ADR data format**

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- The number of microphones.
- The slot width selected.
- The LSBFIRST bit.

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

**SAI_ADR data format example**

**32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to Figure 387.

For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

**Figure 387. SAI_ADR format in TDM, 32-bit slot width**

16-bit slot width (DS = 0b100 and SLOTSZ = 0). Refer to Figure 388.

For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.
Using an 8-bit slot width (DS = 0b010 and SLOTSZ = 0). Refer to Figure 389.
For an 8-microphone configuration, 8 consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.
For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 4 words of 8 bits are required to acquire a byte from 4 microphones and 2 words from 2 microphones.
Figure 389. SAI_ADR format in TDM, 8-bit slot width

### TDM configuration for PDM interface

SAI_A TDM interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the PDM interface as shown in Table 231 to ensure a good connection with the PDM interface.

#### Table 231. TDM settings

<table>
<thead>
<tr>
<th>Bit Fields</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>0b01</td>
<td>Mode must be MASTER receiver.</td>
</tr>
<tr>
<td>PRTCFG</td>
<td>0b00</td>
<td>Free protocol for TDM.</td>
</tr>
<tr>
<td>DS</td>
<td>X</td>
<td>To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See Table 232.</td>
</tr>
<tr>
<td>LSBFIRST</td>
<td>X</td>
<td>This parameter can be used according to the desired data format.</td>
</tr>
<tr>
<td>CKSTR</td>
<td>0</td>
<td>Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.</td>
</tr>
<tr>
<td>MONO</td>
<td>0</td>
<td>Stereo mode.</td>
</tr>
<tr>
<td>FRL</td>
<td>X</td>
<td>To be adjusted according to the number of microphones (MICNBR). See Table 232.</td>
</tr>
<tr>
<td>FSALL</td>
<td>0</td>
<td>Pulse width is one bit clock cycle.</td>
</tr>
</tbody>
</table>
Adjusting the bitstream clock rate

To program the SAI TDM interface properly, the user application must take into account the settings given in Table 231, and follow the sequence below:

1. Adjust the bit clock frequency ($F_{SCK_A}$) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \frac{MICNBR + 1}{2}$$

MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones; see Section 36.6.17)

2. Set the frame length (FRL) using the following formula

$$FRL = 16 \ (MICNBR + 1) - 1$$

3. Configure the slot size (DS) to a multiple of (FRL+1).
Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMMDLY register.

The new delay values become effective after two TDM frames.
36.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 390 shows an AC’97 audio frame structure.

**Figure 390. AC’97 audio frame**

<table>
<thead>
<tr>
<th>FS</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>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDI</td>
<td>Tag</td>
<td>CMD ADDR</td>
<td>CMD DATA</td>
<td>POM FRONT</td>
<td>POM FRONT</td>
<td>LINE1</td>
<td>DAC</td>
<td>POM CENTER</td>
<td>POM LSURR</td>
<td>POM RSURR</td>
<td>POM LFE</td>
<td>LINE2</td>
</tr>
<tr>
<td>SDO</td>
<td>Tag</td>
<td>STATUS ADDR</td>
<td>STATUS DATA</td>
<td>POM LEFT</td>
<td>POM RIGHT</td>
<td>LINE1</td>
<td>ADC</td>
<td>POM MIC</td>
<td>RSR</td>
<td>RSR</td>
<td>RSR</td>
<td>LINE2</td>
</tr>
</tbody>
</table>

Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC’97 protocol.
Clock generator programming in AC’97 mode

In AC’97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in Section 36.4.8: SAI clock generator must be used with FRL = 255, to generate the proper frame rate (F_{FS,x}).

36.4.12 SPDIF output

The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRFCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in Figure 391.

Figure 391. SPDIF format

An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see Table 233). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.
The data stored in SAI_xDR has to be filled as follows:

- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].
If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].
SAI_xDR[23] always represents the MSB.

**Figure 392. SAI_xDR register ordering**

<table>
<thead>
<tr>
<th>SOPD</th>
<th>Preamble coding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>last bit is 0</td>
<td>last bit is 1</td>
</tr>
<tr>
<td>B</td>
<td>11101000</td>
<td>00010111</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channel A data at the start of block</td>
</tr>
<tr>
<td>W</td>
<td>11100100</td>
<td>00011011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channel B data somewhere in the block</td>
</tr>
<tr>
<td>M</td>
<td>11100010</td>
<td>00011101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channel A data</td>
</tr>
</tbody>
</table>

**Note:** The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in **Table 234**.

**Table 234. Parity bit calculation**

<table>
<thead>
<tr>
<th>SAI_xDR[26:0]</th>
<th>Parity bit P value transferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>odd number of 0</td>
<td>0</td>
</tr>
<tr>
<td>odd number of 1</td>
<td>1</td>
</tr>
</tbody>
</table>

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be
executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.
   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

**Clock generator programming in SPDIF generator mode**

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

<table>
<thead>
<tr>
<th>Audio sampling frequencies ( (F_S) )</th>
<th>Symbol-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.1 kHz</td>
<td>2.8224 MHz</td>
</tr>
<tr>
<td>48 kHz</td>
<td>3.072 MHz</td>
</tr>
<tr>
<td>96 kHz</td>
<td>6.144 MHz</td>
</tr>
<tr>
<td>192 kHz</td>
<td>12.288 MHz</td>
</tr>
</tbody>
</table>

More generally, the relationship between the audio sampling frequency \( (F_S) \) and the bit clock rate \( (F_{SCK_x}) \) is given by the formula:

\[
F_S = \frac{F_{SCK_x}}{128}
\]

The bit clock rate is obtained as follows:

\[
F_{SCK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}
\]

*Note:* The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.
36.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the new frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:

1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.
In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

**Companding mode**

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in Figure 393. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1’s complement or 2’s complement representation, depending on the CPL bit setting in the SAI_xCR2 register. In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.
Expansion and compression mode are automatically selected through SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

**Output data line management on an inactive slot**

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).

- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame. *Figure 394* illustrates these behaviors.
When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to Figure 395 (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).
Figure 395. Tristate on output data line in a protocol like I2S

If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in Figure 394 and Figure 395 are replaced by a drive with a value of 0.

36.4.14 Error flags

The SAI implements the following error flags:
- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC’97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to Figure 396).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.
An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to Figure 397). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.
Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that "parasitic" events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in Anticipated frame synchronization detection (AFSDET)).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the signal FS is not used by the protocol.
Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- \((FRL+1)\) is not a power of 2, and
- \((FRL+1)\) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

36.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

36.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to Section 36.4.9: Internal FIFOs). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:
1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

### 36.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in Table 236.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt source</th>
<th>Interrupt group</th>
<th>Audio block mode</th>
<th>Interrupt enable</th>
<th>Interrupt clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ</td>
<td>FREQ</td>
<td>FREQ</td>
<td>Master or slave Receiver or transmitter</td>
<td>FREQIE in SAI_xtM register</td>
<td>Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to Section 36.4.9: Internal FIFOs</td>
</tr>
<tr>
<td>OVRUDR</td>
<td>ERROR</td>
<td>Master or slave Receiver or transmitter</td>
<td>OVRUDRIE in SAI_xtM register</td>
<td>COVRUDR = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
<tr>
<td>AFSDET</td>
<td>ERROR</td>
<td>Slave (not used in AC’97 mode and SPDIF mode)</td>
<td>AFSDETIE in SAI_xtM register</td>
<td>CAFSDET = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
<tr>
<td>LFSDET</td>
<td>ERROR</td>
<td>Slave (not used in AC’97 mode and SPDIF mode)</td>
<td>LFSDETIE in SAI_xtM register</td>
<td>CLFSDET = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
<tr>
<td>CNRDY</td>
<td>ERROR</td>
<td>Slave (only in AC’97 mode)</td>
<td>CNRDYIE in SAI_xtM register</td>
<td>CCNRDY = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
<tr>
<td>MUTEDET</td>
<td>MUTE</td>
<td>Master or slave Receiver mode only</td>
<td>MUTEDETIE in SAI_xtM register</td>
<td>CMUTEDET = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
<tr>
<td>WCKCFG</td>
<td>ERROR</td>
<td>Master with NODIV = 0 in SAI_xCR1 register</td>
<td>WCKCFGIE in SAI_xtM register</td>
<td>CWCKCFG = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
</tbody>
</table>

Follow the sequence below to enable an interrupt:
1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.
### 36.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

#### 36.6.1 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x0004

Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>Bit 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td><strong>MCKEN</strong>: Master clock generation enable</td>
</tr>
<tr>
<td></td>
<td>0: The master clock is not generated</td>
</tr>
<tr>
<td></td>
<td>1: The master clock is generated independently of SAIEN bit</td>
</tr>
<tr>
<td>Bit 26</td>
<td><strong>OSR</strong>: Oversampling ratio for master clock</td>
</tr>
<tr>
<td></td>
<td>This bit is meaningful only when NODIV bit is set to 0.</td>
</tr>
<tr>
<td></td>
<td>0: Master clock frequency = ( F_{FS} \times 256 )</td>
</tr>
<tr>
<td></td>
<td>1: Master clock frequency = ( F_{FS} \times 512 )</td>
</tr>
<tr>
<td>Bits 25:20</td>
<td><strong>MCKDIV[5:0]</strong>: Master clock divider</td>
</tr>
<tr>
<td></td>
<td>These bits are set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>000000: Divides by 1 the kernel clock input (sai_x_ker_ck).</td>
</tr>
<tr>
<td></td>
<td>Otherwise, The master clock frequency is calculated according to the formula given in Section 36.4.8: SAI clock generator.</td>
</tr>
<tr>
<td></td>
<td>These bits have no meaning when the audio block is slave.</td>
</tr>
<tr>
<td></td>
<td>They have to be configured when the audio block is disabled.</td>
</tr>
<tr>
<td>Bit 19</td>
<td><strong>NODIV</strong>: No divider</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512</td>
</tr>
<tr>
<td></td>
<td>1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]</td>
</tr>
<tr>
<td>Bit 18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 17</td>
<td><strong>DMAEN</strong>: DMA enable</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: DMA disabled</td>
</tr>
<tr>
<td></td>
<td>1: DMA enabled</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.</td>
</tr>
</tbody>
</table>
Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.

Note: *When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.*

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.

Note: *This bit has to be set before enabling the audio block and after the audio block configuration.*

Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to Section : Mono/stereo mode for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable
These bits are set and cleared by software. They must be configured when the audio subblock is disabled.
00: audio subblock in asynchronous mode.
01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode
10: Reserved.
11: Reserved.

Note: *The audio subblock must be configured as asynchronous when SPDIF mode is enabled.*

Bit 9 **CKSTR**: Clock strobing edge
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.
0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.
1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST**: Least significant bit first
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.
0: Data are transferred with MSB first
1: Data are transferred with LSB first
Bits 7:5  **DS[2:0]: Data size**

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

- 000: Reserved
- 001: Reserved
- 010: 8 bits
- 011: 10 bits
- 100: 16 bits
- 101: 20 bits
- 110: 24 bits
- 111: 32 bits

Bit 4  **Reserved, must be kept at reset value.**

Bits 3:2  **PRTCFG[1:0]: Protocol configuration**

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

- 00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.
- 01: SPDIF protocol
- 10: AC'97 protocol
- 11: Reserved

Bits 1:0  **MODE[1:0]: SAIx audio block mode**

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

- 00: Master transmitter
- 01: Master receiver
- 10: Slave transmitter
- 11: Slave receiver

*Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).*

### 36.6.2  **SAI configuration register 2 (SAI_ACR2)**

Address offset: 0x0008

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>
</table>

Bits 31:16  **Reserved, must be kept at reset value.**
Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The μ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on CPL bit.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter. The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to **Section : Companding mode** for more details.

- 00: No companding algorithm
- 01: Reserved.
- 10: μ-Law algorithm
- 11: A-Law algorithm

*Note: Companding mode is applicable only when Free protocol mode is selected.*

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software. It defines the type of complement to be used for companding mode.

- 0: 1’s complement representation.
- 1: 2’s complement representation.

*Note: This bit has effect only when the companding mode is μ-Law algorithm or A-Law algorithm.*

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode. The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to **Section : Mute mode** for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It must be written before enabling the audio block: SAIEN. This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to **Section : Mute mode** for more details.

- 0: Bit value 0 is sent during the mute mode.
- 1: Last values are sent during the mute mode.

*Note: This bit is meaningless and must not be used for SPDIF audio blocks.*

Bit 5 **MUTE**: Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to **Section : Mute mode** for more details.

- 0: No mute mode.
- 1: Mute mode enabled.

*Note: This bit is meaningless and must not be used for SPDIF audio blocks.*
Bit 4 **TRIS**: Tristate management on data line.
This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.
Refer to Section: Output data line management on an inactive slot for more details.
0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH**: FIFO flush.
This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.
0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.

Bits 2:0 **FTH[2:0]**: FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

### 36.6.3 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x000C
Reset value: 0x0000 0007

*Note:* This register has no meaning in AC'97 and SPDIF audio protocol.

<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>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>
</tbody>
</table>

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

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.
Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g.: this bit has to be set for I2S or LSB/MSB-justified protocols...).
This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame.
These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

### 36.6.4 SAI slot register (SAI_ASLOTR)
Address offset: 0x0010
Reset value: 0x0000 0000

*Note*: This register has no meaning in AC’97 and SPDIF audio protocol.
Bits 31:16 **SLOTEN[15:0]**: Slot enable.
   These bits are set and cleared by software.
   Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
   0: Inactive slot.
   1: Active slot.
   The slot must be enabled when the audio block is disabled.
   They are ignored in AC’97 or SPDIF mode.

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

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
   These bits are set and cleared by software.
   The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots).
   The maximum number of slots is 16.
   The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
   They are ignored in AC’97 or SPDIF mode.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
   This bits is set and cleared by software.
   The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
   Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
   These bits must be set when the audio block is disabled.
   They are ignored in AC’97 or SPDIF mode.

   00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
   01: 16-bit
   10: 32-bit
   11: Reserved

   Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
   These bits are set and cleared by software.
   The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
   These bits must be set when the audio block is disabled.
   They are ignored in AC’97 or SPDIF mode.

**36.6.5 SAI interrupt mask register (SAI_AIM)**

Address offset: 0x0014

Reset value: 0x0000 0000
Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5  **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4  **CNRDYIE**: Codec not ready interrupt enable (AC’97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3  **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode.

Bit 2  **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
*Note: This bit is used only in Free protocol mode and is meaningless in other modes.*

Bit 1  **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0  **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.
### 36.6.6 SAI status register (SAI_ASR)

Address offset: 0x0018
Reset value: 0x0000 0008

<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>LFSDE T</td>
<td>AFSDET</td>
<td>CNRDY</td>
<td>FREQ</td>
<td>WCKCFG</td>
<td>MUTEDET</td>
<td>OVRUDR</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>

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

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).

000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETE bit is set in the SAI_xIM register.
This flag is cleared when the software sets CLFSDE T in SAI_xCLRFR register.

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if AFSDETE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDE T in SAI_xCLRFR register.
Bit 4 **CNRDY**: Codec not ready.
   This bit is read only.
   0: External AC’97 Codec is ready
   1: External AC’97 Codec is not ready
   This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and
   configured in receiver mode.
   It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
   This bit is read only.
   0: No FIFO request.
   1: FIFO request to read or to write the SAI_xDR.
   The request depends on the audio block configuration:
   – If the block is configured in transmission mode, the FIFO request is related to a write request
     operation in the SAI_xDR.
   – If the block configured in reception, the FIFO request related to a read request operation from the
     SAI_xDR.
   This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
   This bit is read only.
   0: Clock configuration is correct
   1: Clock configuration does not respect the rule concerning the frame length specification defined in
   Section 36.4.6: Frame synchronization (configuration of FRL[7:0] bit in the SAI_xFRCR register)
   This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
   It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
   This bit is read only.
   0: No MUTE detection on the SD input line
   1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio
   frame
   This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a
   consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
   It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
   This bit is read only.
   0: No overrun/underrun error.
   1: Overrun/underrun error detection.
   The overrun and underrun conditions can occur only when the audio block is configured as a
   receiver and a transmitter, respectively.
   It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.
36.6.7 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x001C
Reset value: 0x0000 0000

| Bit 31:7 | Bit 6 CLFSDET: Clear late frame synchronization detection flag. This bit is write only. Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register. This bit is not used in AC’97 or SPDIF mode Reading this bit always returns the value 0. |
| Bit 5 CAFSDET: Clear anticipated frame synchronization detection flag. This bit is write only. Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register. It is not used in AC’97 or SPDIF mode Reading this bit always returns the value 0. |
| Bit 4 CCNRDY: Clear Codec not ready flag. This bit is write only. Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register. This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register. Reading this bit always returns the value 0. |
| Bit 3 Reserved, must be kept at reset value. |
| Bit 2 CWCKCFG: Clear wrong clock configuration flag. This bit is write only. Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register. This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register. Reading this bit always returns the value 0. |
| Bit 1 CMUTEDET: Mute detection flag. This bit is write only. Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register. Reading this bit always returns the value 0. |
| Bit 0 COVRUDR: Clear overrun / underrun. This bit is write only. Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register. Reading this bit always returns the value 0. |
36.6.8 SAI data register (SAI_ADR)

Address offset: 0x0020
Reset value: 0x0000 0000

Bits 31:0 DATA[31:16]: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

36.6.9 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x0024
Reset value: 0x0000 0040

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

Bit 27 MCKEN: Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 OSR: Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_Fs x 256
1: Master clock frequency = F_Fs x 512

Bits 25:20 MCKDIV[5:0]: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in Section 36.4.8: SAI clock generator.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.
Bit 19 **NODIV**: No divider

This bit is set and cleared by software.

0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable

This bit is set and cleared by software.

0: DMA disabled
1: DMA enabled

*Note*: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled
1: SAI audio block enabled.

*Note*: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.

*Note*: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot 1 is discarded and only the data received from slot 0 are stored. Refer to Section: Mono/stereo mode for more details.

0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.
01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode
10: Reserved.
11: Reserved

*Note*: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.
Bit 9 **CKSTR**: Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

- **0**: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.
- **1**: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST**: Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

- **0**: Data are transferred with MSB first
- **1**: Data are transferred with LSB first

Bits 7:5 **DS[2:0]**: Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

- **000**: Reserved
- **001**: Reserved
- **010**: 8 bits
- **011**: 10 bits
- **100**: 16 bits
- **101**: 20 bits
- **110**: 24 bits
- **111**: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]**: Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

- **00**: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.
- **01**: SPDIF protocol
- **10**: AC'97 protocol
- **11**: Reserved

Bits 1:0 **MODE[1:0]**: SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

- **00**: Master transmitter
- **01**: Master receiver
- **10**: Slave transmitter
- **11**: Slave receiver

**Note**: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.
### 36.6.10 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x0028  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15:14</td>
<td>COMP[1:0]: Companding mode.</td>
</tr>
<tr>
<td></td>
<td>These bits are set and cleared by software. The μ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on CPL bit.</td>
</tr>
<tr>
<td></td>
<td>The data expansion or data compression are determined by the state of bit MODE[0].</td>
</tr>
<tr>
<td></td>
<td>The data compression is applied if the audio block is configured as a transmitter.</td>
</tr>
<tr>
<td></td>
<td>The data expansion is automatically applied when the audio block is configured as a receiver.</td>
</tr>
<tr>
<td></td>
<td>Refer to Section: Companding mode for more details.</td>
</tr>
<tr>
<td></td>
<td>00: No companding algorithm</td>
</tr>
<tr>
<td></td>
<td>01: Reserved.</td>
</tr>
<tr>
<td></td>
<td>10: μ-Law algorithm</td>
</tr>
<tr>
<td></td>
<td>11: A-Law algorithm</td>
</tr>
<tr>
<td></td>
<td>Note: Companding mode is applicable only when Free protocol mode is selected.</td>
</tr>
<tr>
<td>Bit 13</td>
<td>CPL: Complement bit.</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>It defines the type of complement to be used for companding mode</td>
</tr>
<tr>
<td></td>
<td>0: 1’s complement representation.</td>
</tr>
<tr>
<td></td>
<td>1: 2’s complement representation.</td>
</tr>
<tr>
<td></td>
<td>Note: This bit has effect only when the companding mode is μ-Law algorithm or A-Law algorithm.</td>
</tr>
<tr>
<td>Bits 12:7</td>
<td>MUTECKNT[5:0]: Mute counter.</td>
</tr>
<tr>
<td></td>
<td>These bits are set and cleared by software. They are used only in reception mode.</td>
</tr>
<tr>
<td></td>
<td>The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.</td>
</tr>
<tr>
<td></td>
<td>Refer to Section: Mute mode for more details.</td>
</tr>
</tbody>
</table>
Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It must be written before enabling the audio block: SAIEN. This bit is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set. If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL. If the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to Section: Mute mode for more details.

0: Bit value 0 is sent during the mute mode.
1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE**: Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to Section: Mute mode for more details.

0: No mute mode.
1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to Section: Output data line management on an inactive slot for more details.

0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH**: FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.

Bits 2:0 **FTH[2:0]**: FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved
36.6.11  SAI frame configuration register (SAI_BFRCR)

Address offset: 0x002C
Reset value: 0x0000 0007

Note:  This register has no meaning in AC’97 and SPDIF audio protocol

<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>r</td>
<td>rw</td>
<td>rw</td>
<td>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 31:19  Reserved, must be kept at reset value.

Bit 18  FSOFF: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17  FSPOL: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16  FSDEF: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15  Reserved, must be kept at reset value.

Bits 14:8  FSALL[6:0]: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.
Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to a value ranging from 8 to 256.

These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.

### 36.6.12 SAI slot register (SAI_BSLOTR)

Address offset: 0x0030

Reset value: 0x0000 0000

**Note:** This register has no meaning in AC’97 and SPDIF audio protocol.

<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>
<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>
</tbody>
</table>

**Bits 31:16 SLOTEN[15:0]:** Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

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

**Bits 11:8 NBSLOT[3:0]:** Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.
Bits 7.6 **SLOTSZ[1:0]**: Slot size

This bit is set and cleared by software.

The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.

Refer to *Output data line management on an inactive slot* for information on how to drive SD line.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).

01: 16-bit

10: 32-bit

11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4.0 **FBOFF[4:0]**: First bit offset

These bits are set and cleared by software.

The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

### 36.6.13 SAI interrupt mask register (SAI_BIM)

Address offset: 0x0034

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>

|                | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.

This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIIE**: Anticipated frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.

This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.
Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC’97).

This bit is set and cleared by software.

0: Interrupt is disabled
1: Interrupt is enabled

When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.

This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled
1: Interrupt is enabled

When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set. Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode.

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled
1: Interrupt is enabled

This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0. It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.

Note: *This bit is used only in Free protocol mode and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled
1: Interrupt is enabled

When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set. This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled
1: Interrupt is enabled

When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

### 36.6.14 SAI status register (SAI_BSR)

Address offset: 0x0038

Reset value: 0x0000 0008
Bits 31:19  Reserved, must be kept at reset value.

Bits 18:16  **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).

- 000: FIFO empty (transmitter and receiver modes)
- 001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
- 010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
- 011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
- 100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
- 101: FIFO full (transmitter and receiver modes)
- Others: Reserved

Bits 15:7  Reserved, must be kept at reset value.

Bit 6  **LFSDET**: Late frame synchronization detection.

This bit is read only.

- 0: No error.
- 1: Frame synchronization signal is not present at the right time.

This flag can be set only if the audio block is configured in slave mode. It is not used in AC’97 or SPDIF mode.

It can generate an interrupt if LFSDET_IE bit is set in the SAI_xIM register.

This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register.

Bit 5  **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.

- 0: No error.
- 1: Frame synchronization signal is detected earlier than expected.

This flag can be set only if the audio block is configured in slave mode. It is not used in AC’97 or SPDIF mode.

It can generate an interrupt if AFSDET_IE bit is set in SAI_xIM register.

This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4  **CNRDY**: Codec not ready.

This bit is read only.

- 0: External AC’97 Codec is ready
- 1: External AC’97 Codec is not ready

This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.

It can generate an interrupt if CNRDY_IE bit is set in SAI_xIM register.

This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3  **FREQ**: FIFO request.

This bit is read only.

- 0: No FIFO request.
- 1: FIFO request to read or to write the SAI_xDR.

The request depends on the audio block configuration:

- If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
- If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQ_IE bit is set in SAI_xIM register.
Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in Section 36.4.6: Frame synchronization (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

### 36.6.15 SAI clear flag register (SAI_BCLRFR)
Address offset: 0x003C
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>
Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 5 **CAFSDDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

### 36.6.16 SAI data register (SAI_BDR)

Address offset: 0x0040

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>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

1270/1530  RM0434 Rev 13
Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

### 36.6.17 SAI PDM control register (SAI_PDMCR)

Address offset: 0x0044
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>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>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>
<td></td>
</tr>
</tbody>
</table>

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

Bits 15:10 Reserved, must be kept at reset value.

- **Bit 9 CKEN2**: Clock enable of bitstream clock number 2
  - This bit is set and cleared by software.
  - 0: SAI_CK2 clock disabled
  - 1: SAI_CK2 clock enabled
  - *Note*: It is not recommended to configure this bit when PDMEN = 1.
  - SAI_CK2 might not be available for all SAI instances. Refer to Section 36.3: SAI implementation for details.

- **Bit 8 CKEN1**: Clock enable of bitstream clock number 1
  - This bit is set and cleared by software.
  - 0: SAI_CK1 clock disabled
  - 1: SAI_CK1 clock enabled
  - *Note*: It is not recommended to configure this bit when PDMEN = 1.
  - SAI_CK1 might not be available for all SAI instances. Refer to Section 36.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.
36.6.18 **SAI PDM delay register (SAI_PDMDLY)**

Address offset: 0x0048
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30-28</th>
<th>Bit 27</th>
<th>Bit 26-24</th>
<th>Bit 23-20</th>
<th>Bit 19-16</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**
This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 T\textsubscript{SAI\_CK} period
- 010: Delay of 2 T\textsubscript{SAI\_CK} periods
- ...
- 111: Delay of 7 T\textsubscript{SAI\_CK} periods

This field can be changed on-the-fly.

*Note: This field can be used only if D4 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 27 Reserved, must be kept at reset value.
Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4

- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 \(T_{SAL\_CK}\) period
- 010: Delay of 2 \(T_{SAL\_CK}\) periods
- ...
- 111: Delay of 7 \(T_{SAL\_CK}\) periods

This field can be changed on-the-fly.

*Note: This field can be used only if D4 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3

- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 \(T_{SAL\_CK}\) period
- 010: Delay of 2 \(T_{SAL\_CK}\) periods
- ...
- 111: Delay of 7 \(T_{SAL\_CK}\) periods

This field can be changed on-the-fly.

*Note: This field can be used only if D3 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3

- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 \(T_{SAL\_CK}\) period
- 010: Delay of 2 \(T_{SAL\_CK}\) periods
- ...
- 111: Delay of 7 \(T_{SAL\_CK}\) periods

This field can be changed on-the-fly.

*Note: This field can be used only if D3 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2

- This bit is set and cleared by software.
- 000: No delay
- 001: Delay of 1 \(T_{SAL\_CK}\) period
- 010: Delay of 2 \(T_{SAL\_CK}\) periods
- ...
- 111: Delay of 7 \(T_{SAL\_CK}\) periods

This field can be changed on-the-fly.

*Note: This field can be used only if D2 line is available. Refer to Section 36.3: SAI implementation to check if it is available.*

Bit 11 Reserved, must be kept at reset value.
Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bit is set and cleared by software.
000: No delay
001: Delay of 1 $T_{SAI}$ periods
010: Delay of 2 $T_{SAI}$ periods
...
111: Delay of 7 $T_{SAI}$ periods

This field can be changed on-the-fly.
*Note:* This field can be used only if D2 line is available. Refer to Section 36.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bit is set and cleared by software.
000: No delay
001: Delay of 1 $T_{SAI}$ period
010: Delay of 2 $T_{SAI}$ periods
...
111: Delay of 7 $T_{SAI}$ periods

This field can be changed on-the-fly.
*Note:* This field can be used only if D1 line is available. Refer to Section 36.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bit is set and cleared by software.
000: No delay
001: Delay of 1 $T_{SAI}$ period
010: Delay of 2 $T_{SAI}$ periods
...
111: Delay of 7 $T_{SAI}$ periods

This field can be changed on-the-fly.
*Note:* This field can be used only if D1 line is available. Refer to Section 36.3: SAI implementation to check if it is available.

### SAI register map

#### Table 237. SAI register map and reset values

| Offset   | Register name | 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  |
|----------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---
Refer to Section 2.2 for the register boundary addresses.
37 Inter-processor communication controller (IPCC)

37.1 Introduction

The inter-processor communication controller (IPCC) is used for communicating data between two processors.

The IPCC block provides a nonblocking signaling mechanism to post and retrieve communication data in an atomic way. It provides the signaling for twelve channels:
- six channels in the direction from processor 1 to processor 2
- six channels in the opposite direction

It is then possible to have two different communication types in each direction.

The IPCC communication data must be located in a common memory, which is not part of the IPCC block.

37.2 IPCC main features

- Status signaling for the twelve channels
  - Channel occupied/free flag, also used as lock
  - Two interrupt lines per processor
    - One for RX channel occupied (communication data posted by sending processor)
    - One for TX channel free (communication data retrieved by receiving processor)

- Interrupt masking per channel
  - Channel occupied mask
  - Channel free mask

- Two channel operation modes
  - Simplex (each channel has its own communication data memory location)
  - Half duplex (a single channel associated to a bidirectional communication data information memory location)

37.3 IPCC functional description

The IPCC communication data is located in a common memory, which is not part of the IPCC block. The address location of the communication data must be known or located in a known common area that, as already stated, is not part of the IPCC block.

For each communication, the IPCC block provides a channel status flag CHnF.

When 0, the channel status flag CHnF indicates that the associated IPCC channel is free (the receiving processor has retrieved communication data), and can be accessed by the sending processor.

When 1, the channel status flag CHnF indicates that the associated IPCC channel is occupied (the sending processor has posted communication data) and can be accessed by the receiving processor.
The channel operation mode must be known to both processors. A common parameter can be used to indicate the channel transfer mode and must also be located in a known common area. This parameter is not available from the IPCC.

### 37.3.1 IPCC block diagram

The IPCC (see Figure 398) consists of the following subblocks:
- Status block, containing the channel status
- IPCC interface block, providing AHB access to the channel status registers
- Interrupt interface block, providing control for the interrupts

![Figure 398. IPCC block diagram](image)

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB slave</td>
<td>I/O</td>
<td>AHB register access bus</td>
</tr>
<tr>
<td>ipcc_tx_free_int1</td>
<td>O</td>
<td>TX free interrupt to processor 1</td>
</tr>
<tr>
<td>ipcc_rx_occupied_int1</td>
<td>O</td>
<td>RX occupied interrupt to processor 1</td>
</tr>
<tr>
<td>ipcc_tx_free_int2</td>
<td>O</td>
<td>TX free interrupt to processor 2</td>
</tr>
<tr>
<td>ipcc_rx_occupied_int2</td>
<td>O</td>
<td>RX occupied interrupt to processor 2</td>
</tr>
</tbody>
</table>

### 37.3.2 IPCC Simplex channel mode

In Simplex channel mode, a dedicated memory location (used to transfer data in a single direction) is assigned to the communication data. The associated channel N control bits (see Table 239) are used to manage the transfer from the sending to the receiving processor.
Once the sending processor has posted the communication data in the memory, it sets the channel status flag CHnF to occupied with CHnS.

Once the receiving processor has retrieved the communication data from the memory, it clears the channel status flag CHnF back to free with CHnC.

**Figure 399. IPCC Simplex channel mode transfer timing**
To send communication data:

The sending processor checks the channel status flag CHnF:

- When CHnF = 0, the channel is free (last communication data retrieved by the receiving processor), and the new communication data can be written.
- When CHnF = 1, the channel is occupied (last communication data not retrieved by the receiving processor), and the sending processor unmasks the channel free interrupt (CHnFM = 0).
- On a TX free interrupt, the sending processor checks which channel became free and masks the channel free interrupt (CHnFM = 1). Then the new communication can take place.

Once the complete communication data is posted, the channel status is set to occupied with CHnS. This gives memory access to the receiving processor and generates the RX occupied interrupt.
To receive a communication, the channel occupied interrupt is unmasked (CHnOM = 0):

On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from memory.

Once the complete communication data is retrieved, the channel status is cleared to free with CHnC. This gives memory access back to the sending processor and may generate the TX free interrupt.

Once the channel status is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

### 37.3.3 IPCC Half-duplex channel mode

The Half-duplex channel mode is used when one processor sends a communication and the other processor sends a response to each communication (ping-pong).

In Half-duplex channel mode, a single dedicated memory location is assigned to communication data and response, and is used to transfer data in both directions. The sending processor channel status flag CHnF is assigned to the channel and used by both processors (see *Table 239*).

Once the processor A posts communication data into memory, it sets the processor A channel status flag CHnF to occupied with CHnS (giving memory access to processor B).

Once the processor B retrieves communication data from memory, it does not change the channel status flags. The memory access is kept by processor B for the response.

Once the processor B posts the response into memory, it clears the channel status flag CHnF to free with CHnC (giving memory access back to processor A).
Once the processor A retrieves the response from the memory, it does not change the channel status flags. The memory location access is kept by processor A for the next communication data.

Figure 402. IPCC Half-duplex channel mode transfer timing

Figure 403. IPCC Half-duplex - Send procedure state diagram
To send communication data:

The sending processor waits for its response pending software variable to get 0.

- Once the response pending software variable is 0 the communication data is posted.

Once the complete communication data has been posted, the channel status flag CHnF is set to occupied with CHnS, and the response pending software variable is set to 1 (this gives memory access and generates the RX occupied interrupt to the receiving processor).

Once the channel status flag CHnF is set, the channel free interrupt is unmasked (CHnFM = 0).

To send a response:

The receiving processor waits for its response pending software variable to get 1.

- Once the response pending software variable is 1, the response is posted.

Once the complete response is posted, the channel status flag CHnF is cleared to free with CHnC, and the response pending software variable is set to 0 (this gives memory access and generates the TX free interrupt to the sending processor).

Once the channel status flag CHnF is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

Figure 404. IPCC Half-duplex - Receive procedure state diagram

Response processor A

Communication processor B

To receive communication data the channel occupied interrupt is unmasked (CHnOM = 0):

On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from the memory.

Once the complete communication data is retrieved, the response pending software variable is set. The channel status is not changed, access to the memory is kept to post the subsequent response.
To receive the response the channel free interrupt is unmasked (CHnFM = 0):

On a TX free interrupt, the sending processor checks which channel became free, masks the associated channel free interrupt (CHnFM) and reads the response from the memory.

Once the complete response is retrieved, the response pending software variable is cleared. The channel status is not changed, access to the memory is kept to post the subsequent communication data.

37.3.4 IPCC interrupts

There are four interrupt lines:

- two RX channel occupied interrupts, one for each processor:
  - Interrupt enable RXOIE per processor
  - Individual mask CHnOM per channel
- two TX channel free interrupts, one for each processor
  - Interrupt enable TXFIE per processor
  - Individual mask CHnFM per channel

The RX occupied interrupt is used by the receiving processor and indicates when an unmasked channel status indicates occupied (CHnF = 1).

The TX free interrupt is used by the sending processor, and indicates when an unmasked channel status indicates free (CHnF = 0).

A secure channel only generates a secure interrupt, and only in the case when the channel is secure unmasked and global secure enabled.

A non-secure channel only generates a non-secure interrupt, and only in the case when the channel is non-secure unmasked and global non-secure enabled.
37.4 IPCC registers

The peripheral registers must be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses are not permitted and do not generate a bus error.

37.4.1 IPCC processor 1 control register (IPCC_C1CR)

Address offset: 0x000
Reset value: 0x0000 0000

| Bit 31:17 | Reserved, must be kept at reset value. |
| Bit 16   | TXFIE: Processor 1 transmit channel free interrupt enable |
|          | Associated with IPCC_C1TOC2SR. |
|          | 1: Enable an unmasked processor 1 transmit channel free to generate a TX free interrupt. |
|          | 0: Processor 1 TX free interrupt disabled |
| Bit 15:1 | Reserved, must be kept at reset value. |
| Bit 0    | RXOIE: Processor 1 receive channel occupied interrupt enable |
|          | Associated with IPCC_C2TOC1SR. |
|          | 1: Enable an unmasked processor 1 receive channel occupied to generate an RX occupied interrupt. |
|          | 0: Processor 1 RX occupied interrupt disabled |

37.4.2 IPCC processor 1 mask register (IPCC_C1MR)

Address offset: 0x004
Reset value: 0xFFFF FFFF

| Bit 31:22 | Reserved, must be kept at reset value. |
37.4.3 IPCC processor 1 status set clear register (IPCC_C1SCR)

Address offset: 0x008

Reset value: 0x0000 0000

Reading this register always 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></td>
<td>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>
<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 31:22 Reserved, must be kept at reset value.

Bits 21:16 CHnS: Processor 1 transmit channel n status set (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Processor 1 transmit channel n status bit set.
0: No action.

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

Bits 5:0 CHnC: Processor 1 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Processor 1 receive channel n status bit clear.
0: No action.
### 37.4.4 IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR)

Address offset: 0x00C  
Reset value: 0x0000 0000

| Bit 31:6 | Reserved, must be kept at reset value. |
| Bit 5:0  | **CHnF**: Processor 1 transmit to processor 2 receive channel n status flag before masking (n = 6 to 1).  
1: Channel occupied, data can be read by the receiving processor 2. Generates a channel RX occupied interrupt to processor 2, when unmasked.  
0: Channel free, data can be written by the sending processor 1. Generates a channel TX free interrupt to processor 1, when unmasked. |

### 37.4.5 IPCC processor 2 control register (IPCC_C2CR)

Address offset: 0x010  
Reset value: 0x0000 0000

| Bit 31:17 | Reserved, must be kept at reset value. |
| Bit 16    | **TXFIE**: Processor 2 transmit channel free interrupt enable  
Associated with IPCC_C2TOC1SR.  
1: Enable an unmasked processor 2 transmit channel free to generate a TX free interrupt.  
0: Processor 2 TX free interrupt disabled |
| Bit 15:1  | Reserved, must be kept at reset value. |
| Bit 0     | **RXOIE**: Processor 2 receive channel occupied interrupt enable  
Associated with IPCC_C1TOC2SR.  
1: Enable an unmasked processor 2 receive channel occupied to generate an RX occupied interrupt.  
0: Processor 2 RX occupied interrupt disabled |
37.4.6 IPCC processor 2 mask register (IPCC_C2MR)

Address offset: 0x014
Reset value: 0xFFFFFFFF

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

Bits 21:16 **CHnFM**: Processor 2 transmit channel n free interrupt mask (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Transmit channel n free interrupt masked.
0: Transmit channel n free interrupt not masked.

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

Bits 5:0 **CHnOM**: Processor 2 receive channel n occupied interrupt mask (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Receive channel n occupied interrupt masked.
0: Receive channel n occupied interrupt not masked.

37.4.7 IPCC processor 2 status set clear register (IPCC_C2SCR)

Address offset: 0x018
Reset value: 0x0000 0000

Reading this register always returns 0x0000 0000.

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

Bits 21:16 **CHnS**: Processor 2 transmit channel n status set (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Processor 2 transmit channel n status bit set.
0: No action.

Bits 15:6 Reserved, must be kept at reset value.
37.4.8 IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR)

Address offset: 0x01C
Reset value: 0x0000 0000

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

Bits 5:0 CHnC: Processor 2 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Processor 2 receive channel n status bit clear.
0: No action.

Bits 5:0 CHnF: Processor 2 transmit to processor 1 receive channel n status flag before masking (n = 6 to 1)
1: Channel occupied, data can be read by the receiving processor 1.
Generates a channel RX occupied interrupt to processor 1, when unmasked.
0: Channel free, data can be written by the sending processor 2.
Generates a channel TX free interrupt to processor 2, when unmasked.
### 37.4.9 IPCC register map

Table 240. IPCC 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  |
|----------|-----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0000   | IPCC_C1CR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0004   | IPCC_C1MR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0008   | IPCC_C1SCR            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x000C   | IPCC_C1TOC2SR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0010   | IPCC_C2CR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0014   | IPCC_C2MR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0018   | IPCC_C2SCR            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x001C   | IPCC_C2TOC1SR         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.2 on page 66 for the register boundary addresses.
38 Hardware semaphore (HSEM)

38.1 Introduction

The hardware semaphore block provides 32 (32-bit) register based semaphores. The semaphores can be used to ensure synchronization between different processes running between different cores. The HSEM provides a non-blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Semaphore lock, in two ways:
  - 2-step lock: by writing COREID and PROCID to the semaphore, followed by a read check
  - 1-step lock: by reading the COREID from the semaphore
- Interrupt generation when a semaphore is unlocked
  - Each semaphore may generate an interrupt on one of the interrupt lines
- Semaphore clear protection
  - A semaphore is only unlocked when COREID and PROCID match
- Global semaphore clear per COREID

38.2 Main features

The HSEM includes the following features:

- 32 (32-bit) semaphores
- 8-bit PROCID
- 4-bit COREID
- One interrupt line per processor
- Lock indication
38.3 Functional description

38.3.1 HSEM block diagram

As shown in Figure 405, the HSEM is based on three sub-blocks:
- the semaphore block containing the semaphore status and IDs
- the semaphore interface block providing AHB access to the semaphore via the HSEM_Rx and HSEM_RLRx registers
- the interrupt interface block providing control for the interrupts via HSEM_CnISR, HSEM_CnIER, HSEM_CnMISR, and HSEM_CnICR registers.

Figure 405. HSEM block diagram

38.3.2 HSEM internal signals

Table 241. HSEM internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB bus</td>
<td>Digital input/output</td>
<td>AHB register access bus</td>
</tr>
<tr>
<td>BusMasterID</td>
<td>Digital input</td>
<td>AHB bus master ID</td>
</tr>
<tr>
<td>hsem_intn_it</td>
<td>Digital output</td>
<td>Interrupt n line (n = 1 to 2)</td>
</tr>
</tbody>
</table>

38.3.3 HSEM lock procedures

There are two lock procedures, namely 2-step (write) lock and 1-step (read) lock. The two procedures can be used concurrently.
The semaphore is free when its LOCK bit is 0. In this case, the COREID and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the COREID indicates which AHB bus master ID has locked it. The PROCID indicates which process of that AHB bus master ID has locked the semaphore.

When write locking a semaphore, the written COREID must match the AHB bus master ID, and the PROCID is written by the AHB bus master software process taking the lock.

When read locking the semaphore, the COREID is taken from the AHB bus master ID, and the PROCID is forced to 0 by hardware. There is no PROCID available with read lock.

**Figure 406. Procedure state diagram**

2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM_Rx register.

- Write semaphore with PROCID and COREID, and LOCK = 1. The COREID data written by software must match the AHB bus master information, that is, a AHB bus master ID = 1 writes data COREID = 1.
- Lock is put in place when the semaphore is free at write time.
- Read-back the semaphore
- The software checks the lock status, if PROCID and COREID match the written data, then the lock is confirmed.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free.

A semaphore can be locked when the PROCID = 0.

Consecutive write attempts with LOCK = 1 to a locked semaphore are ignored.
1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step, carried out from the HSEM_RLRx register.

- Read lock semaphore with the AHB bus master COREID.
- If read COREID matches and PROCID = 0, then lock is put in place. If COREID matches and PROCID is not 0, this means that another process from the same COREID has locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the COREID and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the COREID that locks or locked the semaphore.

**Note:** The 1-step procedure must not be used when running multiple processes of the same AHB bus master ID. All processes using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master ID reads the semaphore as locked by itself with the COREID.

38.3.4 HSEM write/read/read lock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 32-bit semaphore registers, spaced by a 0x80 address offset.

In the first register address bank the semaphore can be written (locked/unlocked) and read through the HSEM_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM_RLRx registers.

38.3.5 HSEM unlock procedures

Unlocking a semaphore is a protected process, to prevent accidental clearing by a AHB bus master ID or by a process not having the semaphore lock right. The procedure consists in writing to the semaphore HSEM_Rx register with the corresponding COREID and PROCID and LOCK = 0. When unlocked the semaphore, the COREID, and the PROCID are all 0.

When unlocked, an interrupt may be generated to signal the event. To this end, the semaphore interrupt must be enabled.

The unlock procedure consists in a write to the semaphore HSEM_Rx register with matching COREID regardless on how the semaphore has been locked (1- or 2-step).

- Write semaphore with PROCID, COREID, and LOCK = 0
- If the written data matches the semaphore PROCID and COREID and the AHB bus master ID, the semaphore is unlocked and an interrupt may be generated when enabled, else write is ignored, semaphore remains locked and no interrupt is generated (the semaphore is locked by another process, AHB bus master ID or the written data does not match the AHB bus master signaling).

**Note:** Different processes of the same AHB bus master ID can write any PROCID value. Preventing other processes of the same AHB bus master ID from unlocking a semaphore must be ensured by software, handling the PROCID correctly.
38.3.6 HSEM COREID semaphore clear

All semaphores locked by a COREID can be unlocked at once by using the HSEM_CR register. Write COREID and correct KEY value in HSEM_CR. All locked semaphores with a matching COREID are unlocked, and may generate an interrupt when enabled.

Note: This procedure may be used in case of an incorrect functioning AHB bus master ID, where another AHB bus master can unlock the locked semaphores by writing the COREID of the incorrect functioning processor into the HSEM_CR register with the correct KEY value. This unlocks all locked semaphores with a matching COREID.

An interrupt may be generated for the unlocked semaphore(s). To this end, the semaphore interrupt must be enabled in the HSEM_CnIER registers.

38.3.7 HSEM interrupts

An interrupt line hsem_intn_it per processor allows each semaphore to generate an interrupt.

An interrupt line provides the following features per semaphore:

- interrupt enable
- interrupt clear
- interrupt status
- masked interrupt status

With the interrupt enable (HSEM_CnIER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status MISF for that semaphore, and do not generate an interrupt on the interrupt line.

The interrupt clear (HSEM_CnICR) clears the interrupt status ISF and masked interrupt status MISF of the associated semaphore for the interrupt line.

The interrupt status (HSEM_CnISR) mirrors the semaphore interrupt status ISF before the enable.

The masked interrupt status (HSEM_CnMISR) only mirrors the semaphore enabled interrupt status MISF on the interrupt line. All masked interrupt status MISF of the enabled semaphores need to be cleared to clear the interrupt line.
The procedure to get an interrupt when a semaphore becomes free is described hereafter.

**Try to lock semaphore x**

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
  - Clear pending semaphore x interrupt status for the interrupt line in HSEM_CnICR.
  - Re-try to lock the semaphore x again:
    - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock and clear semaphore interrupt status).
    - If the semaphore lock fails, enable the semaphore x interrupt in HSEM_CnIER.

**On semaphore x free interrupt, try to lock semaphore x**

- If the semaphore lock is obtained:
  - Disable the semaphore x interrupt in HSEM_CnIER.
  - Clear pending semaphore x interrupt status in HSEM_CnICR.
- If the semaphore x lock fails:
  - Clear pending semaphore x interrupt status in HSEM_CnICR.
  - Try again to lock the semaphore x:
    - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore interrupt status clear), disable the semaphore interrupt in HSEM_CnIER.
– If the semaphore lock fails, wait for semaphore free interrupt.

**Note:** An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.

It is possible to have multiple AHB bus masters informed by the semaphore free interrupts. Each AHB bus master gets its interrupt, and the first one to react locks the semaphore.

### 38.3.8 AHB bus master ID verification

The HSEM allows only authorized AHB bus master IDs to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM_Rx register is checked against the valid bus master IDs.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.

- The AHB bus master 1-step lock read access from the semaphore HSEM_RLRx register is checked against the valid bus master IDs.
  - An unauthorized AHB bus master ID read from HSEM_RLRx returns all 0.

- The semaphore unlock write access to the HSEM_CR register is checked against the valid bus master IDs. Only the valid bus master IDs can write to the HSEM_CR register and unlock any of the COREID semaphores.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not clear the COREID semaphores.

*Table 242* details the relation between bus master/processor and COREID.

<table>
<thead>
<tr>
<th>Bus master 0 (processor1)</th>
<th>Bus master 1 (processor2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COREID = 4</td>
<td>COREID = 8</td>
</tr>
</tbody>
</table>

**Note:** Accesses from unauthorized AHB bus master IDs to other registers are granted.
38.4 HSEM registers

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores, they generate a bus error.

38.4.1 HSEM register semaphore x (HSEM_Rx)

Address offset: 0x0000 + 0x4 * x (x = 0 to 31)
Reset value: 0x0000 0000

The HSEM_Rx must be used to perform a 2-step write lock, read back, and for unlocking a semaphore. Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LOCK: Lock indication</td>
<td>rw</td>
<td>COREID[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>This bit can be written and read by software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: On write free semaphore (only when COREID and PROCID match), on read semaphore is free.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: On write try to lock semaphore, on read semaphore is locked.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-13</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td>PROCID[7:0]</td>
<td></td>
</tr>
<tr>
<td>11-8</td>
<td>COREID[3:0]: Semaphore COREID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Written by software</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- When the semaphore is free and the LOCK bit is at the same time written to 1 and the COREID matches the AHB bus master ID.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched COREID, the COREID is cleared to 0.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- When the semaphore is unlocked (LOCK bit written to 0 or AHB bus master ID does not match COREID, the COREID is not affected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Write when LOCK bit is already 1 (semaphore locked), the COREID is not affected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- An authorized read returns the stored COREID value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7-0</td>
<td>PROCID[7:0]: Semaphore PROCID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Written by software</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-When the semaphore is free and the LOCK is written to 1, and the COREID matches the AHB bus master ID, PROCID is set to the written data.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- When the semaphore is unlocked, LOCK written to 0 and AHB bus master ID matched COREID, the PROCID is cleared to 0.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- When the semaphore is unlocked, LOCK bit written to 0 and AHB bus master ID does not match COREID, the PROCID is not affected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- An authorized read returns the stored PROCID value.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
38.4.2 HSEM read lock register semaphore x (HSEM_RLRx)

Address offset: 0x080 + 0x4 * x (x = 0 to 31)
Reset value: 0x0000 0000

Accesses the same physical bits as HSEM_Rx. The HSEM_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master IDs are granted. Read accesses with unauthorized AHB bus master IDs are discarded and return 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></td>
<td></td>
<td></td>
<td></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></td>
</tr>
</tbody>
</table>

Bit 31 **LOCK**: Lock indication
- This bit is read only by software at this address.
  - When the semaphore is free:
    A read with a valid AHB bus master ID locks the semaphore and returns 1.
  - When the semaphore is locked:
    A read with a valid AHB bus master ID returns 1 (the COREID and PROCID reflect the already locked semaphore information).

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]**: Semaphore COREID
- This field is read only by software at this address.
  - On a read, when the semaphore is free, the hardware sets the COREID to the AHB bus master ID reading the semaphore. The COREID of the AHB bus master locking the semaphore is read.
  - On a read when the semaphore is locked, this field returns the COREID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]**: Semaphore processor ID
- This field is read only by software at this address.
  - On a read when the semaphore is free:
    A read with a valid AHB bus master ID locks the semaphore and hardware sets the PROCID to 0.
  - When the semaphore is locked:
    A read with a valid AHB bus master ID returns the PROCID of the AHB bus master that has locked the semaphore.
38.4.3  **HSEM interrupt enable register (HSEM_CnIER)**

Address offset: \(0x100 + 0x010 \times (n - 1), (n = 1 \text{ to } 2)\)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>ISE[31:16]</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

Bits 31:0  **ISE[31:0]**: Interrupt(n) semaphore x enable bit (x = 0 to 31)

This bit is read and written by software.

0: Interrupt(n) generation for semaphore x disabled (masked)

1: Interrupt(n) generation for semaphore x enabled (not masked)

38.4.4  **HSEM interrupt clear register (HSEM_CnICR)**

Address offset: \(0x104 + 0x010 \times (n - 1), (n = 1 \text{ to } 2)\)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>ISC[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0  **ISC[31:0]**: Interrupt(n) semaphore x clear bit (x = 0 to 31)

This bit is written by software, and is always read 0.

0: Interrupt(n) semaphore x status ISFx and masked status MISFx not affected.

1: Interrupt(n) semaphore x status ISFx and masked status MISFx cleared.

38.4.5  **HSEM interrupt status register (HSEM_CnISR)**

Address offset: \(0x108 + 0x010 \times (n - 1), (n = 1 \text{ to } 2)\)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>ISF[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0  **ISF[31:0]**: Interrupt(n) semaphore x status bit (x = 0 to 31)

This bit is read only.

0: Interrupt(n) semaphore x status ISFx and masked status MISFx not affected.

1: Interrupt(n) semaphore x status ISFx and masked status MISFx cleared.
38.4.6 **HSEM interrupt status register (HSEM_CnMISR)**

Address offset: 0x10C + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:0 ISF[31:0]</th>
<th>Interrupt semaphore x status bit before enable (mask) (x = 0 to 31)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set by hardware, and reset only by software. This bit is cleared by software writing the corresponding HSEM_CnICR bit.</td>
</tr>
<tr>
<td></td>
<td>0: Interrupt semaphore x status, no interrupt pending</td>
</tr>
<tr>
<td></td>
<td>1: Interrupt semaphore x status, interrupt pending</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31:0 MISIF[31:16]</th>
<th>Masked interrupt(n) semaphore x status bit after enable (mask) (x = 0 to 31)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set by hardware and read only by software. This bit is cleared by software writing the corresponding HSEM_CnICR bit. This bit is read as 0 when semaphore x status is masked in HSEM_CnIER bit x.</td>
</tr>
<tr>
<td></td>
<td>0: interrupt(n) semaphore x status after masking not pending</td>
</tr>
<tr>
<td></td>
<td>1: interrupt(n) semaphore x status after masking pending</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31:16 KEY[15:0]</th>
<th>Semaphore clear key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field can be written by software and is always read 0.</td>
</tr>
<tr>
<td></td>
<td>If this key value does not match HSEM_KEYR.KEY, semaphores are not affected.</td>
</tr>
<tr>
<td></td>
<td>If this key value matches HSEM_KEYR.KEY, all semaphores matching the COREID are cleared to the free state.</td>
</tr>
</tbody>
</table>

| Bit 15:13 Reserved, must be kept at reset value. |
| Bit 12 Reserved, must be kept at reset value. |
Bits 11:8 **COREID[3:0]**: COREID of semaphores to be cleared
This field can be written by software and is always read 0.
This field indicates the COREID for which the semaphores are cleared when writing the HSEM_CR.

Bits 7:0 Reserved, must be kept at reset value.

### 38.4.8 **HSEM clear semaphore key register (HSEM_KEYR)**

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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 31:16 **KEY[15:0]**: Semaphore clear key
This field can be written and read by software.
Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.
### 38.4.9 HSEM register map

Table 243. HSEM 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  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | HSEM_R0       | LOCK |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x004  | HSEM_R1       | LOCK |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x07C  | HSEM_R31      | LOCK |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x080  | HSEM_RLR0     | LOCK |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x084  | HSEM_RLR1     | LOCK |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0FC  | HSEM_RLR31    | LOCK |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x100  | HSEM_C1IER    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x104  | HSEM_C1ICR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x108  | HSEM_C1ISR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x10C  | HSEM_C1MISR   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x110  | HSEM_C2IER    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x114  | HSEM_C2ICR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x118  | HSEM_C2ISR    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Resel value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
Table 243. HSEM 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   |
|---------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x11C   | HSEM_C2MISR   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         |               | Resel 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 |
| 0x140   | HSEM_CR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         |               | Resel 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 |
| 0x144   | HSEM_KEYR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|         |               | Resel 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 66 for the register boundary addresses.
39 Universal serial bus full-speed device interface (USB)

39.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported, which allows to stop the device clocks for low-power consumption.

39.2 USB main features

- USB specification version 2.0 full-speed compliant
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 1024 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support
- Battery Charging Specification Revision 1.2 support
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

39.3 USB implementation

Table 244 describes the USB implementation in the devices.

<table>
<thead>
<tr>
<th>USB features(1)</th>
<th>USB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of endpoints</td>
<td>8</td>
</tr>
<tr>
<td>Size of dedicated packet buffer memory SRAM</td>
<td>1024 bytes</td>
</tr>
<tr>
<td>Dedicated packet buffer memory SRAM access scheme</td>
<td>2 x 16 bits / word</td>
</tr>
<tr>
<td>USB 2.0 Link Power Management (LPM) support</td>
<td>X</td>
</tr>
<tr>
<td>Battery Charging Detection (BCD) support</td>
<td>X</td>
</tr>
<tr>
<td>Embedded pull-up resistor on USB_DP line</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X= supported
39.4 **USB functional description**

*Figure 408* shows the block diagram of the USB peripheral.

**Figure 408. USB peripheral block diagram**

The USB peripheral provides an USB-compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 1024 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the USB host, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a buffer description block indicating where the endpoint-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes
The data buffered by the USB peripheral is loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which allows to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to allow the system to immediately restart the normal clock generation and/or support direct clock start/stop.

39.4.1 Description of USB blocks

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- **USB Physical Interface (USB PHY):** This block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for Battery Charging Detection (BCD), multiplexed on same USB_DP and USB_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.

- **Serial Interface Engine (SIE):** The functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as Start of Frame (SOF), USB_Reset, Data errors etc. and to Endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.

- **Timer:** This block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.

- **Packet Buffer Interface:** This block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the Endpoint registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
Endpoint-Related Registers: Each endpoint has an associated register containing the endpoint type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.

Control Registers: These are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.

Interrupt Registers: These contain the Interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

* Endpoint 0 is always used for control transfer in single-buffer mode.

The USB peripheral is connected to the APB1 bus through an APB1 interface, containing the following blocks:

Packet Memory: This is the local memory that physically contains the Packet Buffers. It can be used by the Packet Buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the Packet Memory is 1024 bytes, structured as 512 half-words of 16 bits.

Arbiter: This block accepts memory requests coming from the APB1 bus and from the USB interface. It resolves the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.

Register Mapper: This block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 16-bit wide half-word set addressed by the APB1.

APB1 Wrapper: This provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.

Interrupt Mapper: This block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

### 39.5 Programming considerations

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

#### 39.5.1 Generic USB device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and Isochronous transfers. Apart from system reset, action is always initiated by the USB peripheral, driven by one of the USB events described below.
39.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software should perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{\text{STARTUP}}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the FRES bit in the CNTR register). Clearing the ISTR register then removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB reset (RESET interrupt)

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral will not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the Enable Function (EF) bit of the USB_DADDR register and initializing the EP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RESET interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of reset sequence which triggered the interrupt.

Structure and usage of packet buffers

Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has to be accessed by the microcontroller also, an arbitration logic takes care of the access conflicts, using half APB1 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both the agents can operate as if the packet memory is a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing
back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB1 bus. Different clock configurations are possible where the APB1 clock frequency can be higher or lower than the USB peripheral one.

Note: Due to USB data rate and packet memory interface requirements, the APB1 clock must have a minimum frequency of 10 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory at the address indicated by the USB_BTABL register. Each table entry is associated to an endpoint register and it is composed of four 16-bit half-words so that table start address must always be aligned to an 8-byte boundary (the lowest three bits of USB_BTABL register are always “000”). Buffer descriptor table entries are described in the Section 39.6.2: Buffer descriptor table. If an endpoint is unidirectional and it is neither an Isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to Section 39.5.4: Isochronous transfers and Section 39.5.3: Double-buffered endpoints respectively). The relationship between buffer description table entries and packet buffer areas is depicted in Figure 409.

Figure 409. Packet buffer areas with examples of buffer description table locations

<table>
<thead>
<tr>
<th>Buffer description table locations</th>
<th>Packet buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001_1110 (1E) COUNT3_TX_1</td>
<td>Buffer for double-buffered IN Endpoint 3</td>
</tr>
<tr>
<td>0001_1100 (1C) ADDR3_TX_1</td>
<td></td>
</tr>
<tr>
<td>0001_1010 (1A) COUNT3_TX_0</td>
<td>Buffer for double-buffered OUT Endpoint 2</td>
</tr>
<tr>
<td>0001_1000 (18) ADDR3_TX_0</td>
<td></td>
</tr>
<tr>
<td>0001_0110 (16) COUNT2_RX_1</td>
<td>Transmission buffer for single-buffered Endpoint 1</td>
</tr>
<tr>
<td>0001_0100 (14) ADDR2_RX_1</td>
<td></td>
</tr>
<tr>
<td>0001_0010 (12) COUNT2_RX_0</td>
<td></td>
</tr>
<tr>
<td>0001_0000 (10) ADDR2_RX_0</td>
<td></td>
</tr>
<tr>
<td>0000_1110 (0E) COUNT1_RX</td>
<td>Reception buffer for Endpoint 0</td>
</tr>
<tr>
<td>0000_1100 (0C) ADDR1_RX</td>
<td></td>
</tr>
<tr>
<td>0000_1010 (0A) COUNT1_TX</td>
<td>Transmission buffer for Endpoint 0</td>
</tr>
<tr>
<td>0000_1000 (08) ADDR1_TX</td>
<td></td>
</tr>
<tr>
<td>0000_0110 (06) COUNT0_RX</td>
<td></td>
</tr>
<tr>
<td>0000_0100 (04) ADDR0_RX</td>
<td></td>
</tr>
<tr>
<td>0000_0010 (02) COUNT0_TX</td>
<td></td>
</tr>
<tr>
<td>0000_0000 (00) ADDR0_TX</td>
<td></td>
</tr>
</tbody>
</table>
Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data are copied to the memory only up to the last available location.

**Endpoint initialization**

The first step to initialize an endpoint is to write appropriate values to the ADDRxn_TX/ADDRxn_RX registers so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The EP_TYPE bits in the USB_EPnR register must be set according to the endpoint type, eventually using the EP_KIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STAT_TX bits in the USB_EPnR register and COUNTnx_TX must be initialized. For reception, STAT_RX bits must be set to enable reception and COUNTnx_RX must be written with the allocated buffer size using the BL_SIZE and NUM_BLOCK fields. Unidirectional endpoints, except Isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, registers USB_EPnR and locations ADDRxn_TX/ADDRxn_RX, COUNTnx_TX/COUNTnx_RX (respectively), should not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

**IN packets (data transmission)**

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of ADDRxn_TX and COUNTnx_TX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16 bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (Refer to *Structure and usage of packet buffers on page 1308*) and starts sending a DATA0 or DATA1 PID according to USB_EPnR bit DTOG_TX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STAT_TX bits in the USB_EPnR register.

The ADDR internal register is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRxn_TX for COUNTnx_TX/2 half-words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_EPnR register is updated in the following way: DTOG_TX bit is toggled, the endpoint is made invalid by setting STAT_TX=10 (NAK) and bit CTR_TX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. Servicing of the CTR_TX event starts clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTnx_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STAT_TX to ‘11 (VALID) to re-enable transmissions. While the STAT_TX bits are equal to ‘10 (NAK), any IN request addressed to that endpoint is NAKed,
indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

OUT and SETUP packets (data reception)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX is stored directly in its internal register ADDR. While COUNT is now reset and the values of BL_SIZE and NUM_BLOCK bit fields, which are read within COUNTn_RX content are used to initialize BUF_COUNT, an internal 16 bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STAT_RX in the USB_EPnR register and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, CRC included (i.e. data payload length + 2), or up to the last allocated memory location, as defined by BL_SIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BL_SIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_EPnR register is updated in the following way: DTOG_RX bit is toggled, the endpoint is made invalid by setting STAT_RX = ‘10 (NAK) and bit CTR_RX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. The CTR_RX event is serviced by first determining the transaction type (SETUP bit in the USB_EPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being
processed. After the received data is processed, the application software should set the STAT_RX bits to ‘11 (Valid) in the USB_EPnR, enabling further transactions. While the STAT_RX bits are equal to ‘10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

**Control transfers**

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOG_TX and DTOG_RX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STAT_TX and STAT_RX are set to ‘10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_EPnR register at each CTR_RX event to distinguish normal OUT transactions from SETUP ones. A USB device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction should be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction should be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software will change NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EP_KIND in the USB_EPnR register) bit should be set, so that an error is generated if a status transaction is performed with not-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STAT_RX to VALID (to accept a new command) and STAT_TX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic doesn’t allow a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STAT_RX bits are set to ‘01 (STALL) or ‘10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued CTR_RX request not yet acknowledged by the application (i.e. CTR_RX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the CTR_RX interrupt.
Double-buffered endpoints

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it will answer with a NAK handshake and the host PC will issue the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_EPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from '00 (Disabled): STAT_RX if the double-buffered bulk endpoint is enabled for reception, STAT_TX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOG_RX (bit 14 of USB_EPnR register) for ‘reception’ double-buffered bulk endpoints or DTOG_TX (bit 6 of USB_EPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_EPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_EPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.
The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

**Table 245. Double-buffering buffer flag definition**

<table>
<thead>
<tr>
<th>Buffer flag</th>
<th>‘Transmission’ endpoint</th>
<th>‘Reception’ endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTOG</td>
<td>DTOG_TX (USB_EPnR bit 6)</td>
<td>DTOG_RX (USB_EPnR bit 14)</td>
</tr>
<tr>
<td>SW_BUF</td>
<td>USB_EPnR bit 14</td>
<td>USB_EPnR bit 6</td>
</tr>
</tbody>
</table>

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after

**Table 246. Bulk double-buffering memory buffers usage**

<table>
<thead>
<tr>
<th>Endpoint type</th>
<th>DTOG</th>
<th>SW_BUF</th>
<th>Packet buffer used by USB peripheral</th>
<th>Packet buffer used by Application Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>0</td>
<td>1</td>
<td>ADDRn_TX_0 / COUNTn_TX_0</td>
<td>ADDRn_TX_1 / COUNTn_TX_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>ADDRn_TX_1 / COUNTn_TX_1</td>
<td>ADDRn_TX_0 / COUNTn_TX_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>None (1)</td>
<td>ADDRn_TX_0 / COUNTn_TX_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>None (1)</td>
<td>ADDRn_TX_1 / COUNTn_TX_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td>OUT</td>
<td>0</td>
<td>1</td>
<td>ADDRn_RX_0 / COUNTn_RX_0</td>
<td>ADDRn_RX_1 / COUNTn_RX_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>ADDRn_RX_1 / COUNTn_RX_1</td>
<td>ADDRn_RX_0 / COUNTn_RX_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>None (1)</td>
<td>ADDRn_RX_0 / COUNTn_RX_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>None (1)</td>
<td>ADDRn_RX_1 / COUNTn_RX_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Buffer description table locations.</td>
<td>Buffer description table locations.</td>
</tr>
</tbody>
</table>

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by:

- Writing EP_TYPE bit field at ‘00 in its USB_EPnR register, to define the endpoint as a bulk, and
- Setting EP_KIND bit at ‘1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after
DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains ‘11 (Valid). However, as the token packet of a new transaction is received, the actual endpoint status is masked as ‘10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see Table 246 on page 1314). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing ‘1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control will take place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from ‘11 (Valid) into the STAT bit pair of the related USB_EPnR register. In this case, the USB peripheral will always use the programmed endpoint status, regardless of the buffer usage condition.

39.5.4 Isochronous transfers

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘Isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be 'isochronous' during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for Isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, Isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The Isochronous behavior for an endpoint is selected by setting the EP_TYPE bits at ‘10 in its USB_EPnR register; since there is no handshake phase the only legal values for the STAT_RX/STAT_TX bit pairs are ‘00 (Disabled) and ‘11 (Valid), any other value will produce results not compliant to USB standard. Isochronous endpoints implement double-buffering to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOG_RX for ‘reception’ isochronous endpoints, DTOG_TX for ‘transmission’ isochronous endpoints, both in the related USB_EPnR register) according to Table 247.
As it happens with double-buffered bulk endpoints, the USB_EPnR registers used to implement Isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have Isochronous endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for Isochronous transfers due to the lack of handshake phase, the endpoint remains always ‘11 (Valid). CRC errors or buffer-overrun conditions occurring during Isochronous OUT transfers are anyway considered as correct transactions and they always trigger an CTR_RX event. However, CRC errors will anyway set the ERR bit in the USB_ISTR register to notify the software of the possible data corruption.

### 39.5.5 Suspend/Resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to ‘1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.
The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the FSUSP bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set LP_MODE bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the wakeing event is an USB reset sequence (See “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the LP_MODE bit in USB_CNTR register asynchronously. Even if this event can trigger an WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure should address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear FSUSP bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to Table 248, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the Idle bus state; moreover at the end of a reset sequence the RESET bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which should be handled as usual.

<table>
<thead>
<tr>
<th>[RXDP,RXDM] status</th>
<th>Wake-up event</th>
<th>Required resume software action</th>
</tr>
</thead>
<tbody>
<tr>
<td>“00”</td>
<td>Root reset</td>
<td>None</td>
</tr>
<tr>
<td>“10”</td>
<td>None (noise on bus)</td>
<td>Go back in Suspend mode</td>
</tr>
</tbody>
</table>

Table 248. Resume event detection
A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (e.g. a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the RESUME bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the RESUME bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

**Note:** The RESUME bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the FSUSP bit in USB_CNTR register to 1.

### Table 248. Resume event detection (continued)

<table>
<thead>
<tr>
<th>[RXDP,RXDM] status</th>
<th>Wake-up event</th>
<th>Required resume software action</th>
</tr>
</thead>
<tbody>
<tr>
<td>“01”</td>
<td>Root resume</td>
<td>None</td>
</tr>
<tr>
<td>“11”</td>
<td>Not allowed (noise on bus)</td>
<td>Go back in Suspend mode</td>
</tr>
</tbody>
</table>
39.6 USB and USB SRAM registers

The USB peripheral registers can be divided into the following groups:
- Common Registers: Interrupt and Control registers
- Endpoint Registers: Endpoint configuration and status

The USB SRAM registers cover:
- Buffer Descriptor Table: Location of packet memory used to locate data buffers (see Section 2.2: Memory organization to find USB SRAM base address).

All register addresses are expressed as offsets with respect to the USB peripheral registers base address, except the buffer descriptor table locations, which starts at the USB SRAM base address offset by the value specified in the USB_BTABLE register.

Refer to Section 1.2 on page 61 for a list of abbreviations used in register descriptions.

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

39.6.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

**USB control register (USB_CNTR)**

Address offset: 0x40
Reset value: 0x0003

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>CTRM: Correct transfer interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: Correct Transfer (CTR) Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: CTR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.</td>
</tr>
<tr>
<td>14</td>
<td>PMAOVRM: Packet memory area over / underrun interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: PMAOVR Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: PMAOVR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.</td>
</tr>
<tr>
<td>13</td>
<td>ERRM: Error interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: ERR Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: ERR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.</td>
</tr>
<tr>
<td>12</td>
<td>WKUPM: Wake-up interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: WKUP Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: WKUP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.</td>
</tr>
</tbody>
</table>
Bit 11 **SUSPM**: Suspend mode interrupt mask
   0: Suspend Mode Request (SUSP) Interrupt disabled.
   1: SUSP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RESETM**: USB reset interrupt mask
   0: RESET Interrupt disabled.
   1: RESET Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM**: Start of frame interrupt mask
   0: SOF Interrupt disabled.
   1: SOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM**: Expected start of frame interrupt mask
   0: Expected Start of Frame (ESOF) Interrupt disabled.
   1: ESOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 **L1REQM**: LPM L1 state request interrupt mask
   0: LPM L1 state request (L1REQ) Interrupt disabled.
   1: L1REQ Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **L1RESUME**: LPM L1 Resume request
   The microcontroller can set this bit to send a LPM L1 Resume signal to the host. After the signaling ends, this bit is cleared by hardware.

Bit 4 **RESUME**: Resume request
   The microcontroller can set this bit to send a Resume signal to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the Host PC is ready to drive the resume sequence up to its end.

Bit 3 **FSUSP**: Force suspend
   Software must set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms.
   0: No effect.
   1: Enter suspend mode. Clocks and static power dissipation in the analog transceiver are left unaffected. If suspend power consumption is a requirement (bus-powered device), the application software should set the LP_MODE bit after FSUSP as explained below.

Bit 2 **LP_MODE**: Low-power mode
   This mode is used when the suspend-mode power constraints require that all static power dissipation is avoided, except the one required to supply the external pull-up resistor. This condition should be entered when the application is ready to stop all system clocks, or reduce their frequency in order to meet the power consumption requirements of the USB suspend condition. The USB activity during the suspend mode (WKUP event) asynchronously resets this bit (it can also be reset by software).
   0: No Low-power mode.
   1: Enter Low-power mode.
USB interrupt status register (USB_ISTR)

Address offset: 0x44
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>CTR</td>
<td>This bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_EPnR register. For endpoint-related interrupts, the software can use the Direction of Transaction (DIR) and EP_ID read-only bits to identify, which endpoint made the last interrupt request and called the corresponding interrupt service routine.</td>
</tr>
<tr>
<td>14</td>
<td>PMA OVR</td>
<td>This bit can be cleared by a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with ‘0’ (these bits can only be cleared by software). Read-modify-write cycles should be avoided because between the read and the write operations another bit is read.</td>
</tr>
<tr>
<td>13</td>
<td>ERR</td>
<td>This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.</td>
</tr>
<tr>
<td>12</td>
<td>WKUP</td>
<td>0: Exiting Power Down. 1: Entering Power down mode.</td>
</tr>
<tr>
<td>11</td>
<td>SUSP</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>10</td>
<td>RESET</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>9</td>
<td>SOF</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>8</td>
<td>ESOF</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>7</td>
<td>L1REQ</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>6</td>
<td>Res</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>5</td>
<td>Res</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>4</td>
<td>DIR</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
<tr>
<td>3</td>
<td>EP_ID[3:0]</td>
<td>0: Clear USB reset. 1: Force a reset of the USB peripheral, exactly like a RESET signaling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.</td>
</tr>
</tbody>
</table>

This register contains the status of all the interrupt sources allowing application software to determine, which events caused an interrupt request.

Endpoint transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_EPnR register (the CTR bit is actually a read only bit). For endpoint-related interrupts, the software can use the Direction of Transaction (DIR) and EP_ID read-only bits to identify, which endpoint made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with ‘0’ (these bits can only be cleared by software). Read-modify-write cycles should be avoided because between the read and the write operations another bit
could be set by the hardware and the next write will clear it before the microprocessor has the time to serve the event.

The following describes each bit in detail:

**Bit 15  CTR:** Correct transfer
This bit is set by the hardware to indicate that an endpoint has successfully completed a transaction; using DIR and EP_ID bits software can determine which endpoint requested the interrupt. This bit is read-only.

**Bit 14  PMAOVR:** Packet memory area over / underrun
This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host will retry the transaction. The PMAOVR interrupt should never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during Isochronous transfers (no Isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 13  ERR:** Error
This flag is set whenever one of the errors listed below has occurred:
NANS: No ANSwer. The timeout for a host response has expired.
CRC: Cyclic Redundancy Check error. One of the received CRCs, either in the token or in the data, was wrong.
BST: Bit Stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.
FVIO: Framing format Violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).
The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (e.g. loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 12  WKUP:** Wake-up
This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the LP_MODE bit in the CTRL register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (e.g. wake-up unit) about the start of the resume process. This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 11  SUSP:** Suspend mode request
This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (FSUSP=1) until the end of resume sequence. This bit is read/write but only '0 can be written and writing '1 has no effect.
Bit 10  **RESET**: USB reset request

Set when the USB peripheral detects an active USB RESET signal at its inputs. The USB peripheral, in response to a RESET, just resets its internal protocol state machine, generating an interrupt if RESETM enable bit in the USB_CNTR register is set. Reception and transmission are disabled until the RESET bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RESET interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event. This bit is read/write but only ’0 can be written and writing ’1 has no effect.

Bit 9  **SOF**: Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this could be useful for isochronous applications). This bit is read/write but only ’0 can be written and writing ’1 has no effect.

Bit 8  **ESOF**: Expected start of frame

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the Suspend Timer issues this interrupt. If three consecutive ESOF interrupts are generated (i.e. three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the Suspend Timer is not yet locked. This bit is read/write but only ’0 can be written and writing ’1 has no effect.

Bit 7  **L1REQ**: LPM L1 state request

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only ’0 can be written and writing ’1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4  **DIR**: Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request. If DIR bit=0, CTR_TX bit is set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC). If DIR bit=1, CTR_RX bit or both CTR_TX/CTR_RX are set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB_EPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.
Bits 3:0  **EP_ID[3:0]: Endpoint Identifier**  
These bits are written by the hardware according to the endpoint number, which generated the interrupt request. If several endpoint transactions are pending, the hardware writes the endpoint identifier related to the endpoint having the highest priority defined in the following way: Two endpoint sets are defined, in order of priority: Isochronous and double-buffered bulk endpoints are considered first and then the other endpoints are examined. If more than one endpoint from the same set is requesting an interrupt, the EP_ID bits in USBISTR register are assigned according to the lowest requesting endpoint register, EP0R having the highest priority followed by EP1R and so on. The application software can assign a register to each endpoint according to this priority scheme, so as to order the concurring endpoint requests in a suitable way. These bits are read only.

**USB frame number register (USB_FNR)**

Address offset: 0x48  
Reset value: 0x0XXX where X is undefined

| Bit 15 | RXDP: Receive data + line status  
This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event. |
| Bit 14 | RXDM: Receive data - line status  
This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event. |
| Bit 13 | LCK: Locked  
This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs. |
| Bits 12:11 | LSOF[1:0]: Lost SOF  
These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared. |
| Bits 10:0 | FN[10:0]: Frame number  
This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for Isochronous transfers. This bit field is updated on the generation of an SOF interrupt. |

**USB device address (USB_DADDR)**

Address offset: 0x4C  
Reset value: 0x0000
**Buffer table address (USB_BTABLE)**

Address offset: 0x50  
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  

**Bit 7**  **EF**: Enable function  
This bit is set by the software to enable the USB device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at '0' no transactions are handled, irrespective of the settings of USB_EPnR registers.

**Bits 6:0**  **ADD[6:0]**: Device address  
These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the Endpoint Address (EA) field in the associated USB_EPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

**LPM control and status register (USB_LPMCSR)**

Address offset: 0x54  
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 15:3**  **BTABLE[15:3]**: Buffer table  
These bits contain the start address of the buffer allocation table inside the dedicated packet memory. This table describes each endpoint buffer location and size and it must be aligned to an 8 byte boundary (the 3 least significant bits are always '0'). At the beginning of every transaction addressed to this device, the USB peripheral reads the element of this table related to the addressed endpoint, to get its buffer start location and the buffer size (Refer to Structure and usage of packet buffers on page 1308).

**Bits 2:0**  Reserved, forced by hardware to 0.
Battery charging detector (USB_BCDR)

Address offset: 0x58
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>DPPU: DP pull-up control</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value</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>Reserved, must be kept at reset value</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>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value</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>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value</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>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
</tbody>
</table>

Bit 15 **DPPU**: DP pull-up control
This bit is set by software to enable the embedded pull-up on the DP line. Clearing it to ‘0’ can be used to signalize disconnect to the host when needed by the user software.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET**: DM pull-up detection status
This bit is active only during PD and gives the result of comparison between DM voltage level and \(V_{LGC}\) threshold. In normal situation, the DM level should be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.
0: Normal port detected (connected to SDP, ACA, CDP or DCP).
1: PS2 port or proprietary charger detected.

Bit 6 **SDET**: Secondary detection (SD) status
This bit gives the result of SD.
0: CDP detected.
1: DCP detected.

Bit 5 **PDET**: Primary detection (PD) status
This bit gives the result of PD.
0: no BCD support detected (connected to SDP or proprietary device).
1: BCD support detected (connected to ACA, CDP or DCP).
Bit 4 **DCDET**: Data contact detection (DCD) status
This bit gives the result of DCD.
0: data lines contact not detected.
1: data lines contact detected.

Bit 3 **SDEN**: Secondary detection (SD) mode enable
This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 2 **PDEN**: Primary detection (PD) mode enable
This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 1 **DCDEN**: Data contact detection (DCD) mode enable
This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly.

Bit 0 **BCDEN**: Battery charging detector (BCD) enable
This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD should be placed in OFF mode by clearing this bit to '0' in order to allow the normal USB operation.

### Endpoint-specific registers

The number of these registers varies according to the number of endpoints that the USB peripheral is designed to handle. The USB peripheral supports up to 8 bidirectional endpoints. Each USB device must support a control endpoint whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint number value. For each endpoint, an USB_EPnR register is available to store the endpoint specific information.

**USB endpoint n register (USB_EPnR), n=[0..7]**

Address offset: 0x000 to 0x1C

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><strong>CTR_RX</strong></td>
<td><strong>DTOG_RX</strong></td>
<td><strong>STAT_RX[1:0]</strong></td>
<td><strong>SETUP</strong></td>
<td><strong>TYPE[1:0]</strong></td>
<td><strong>EP_KIND</strong></td>
<td><strong>CTR_TX</strong></td>
<td><strong>DTOG_TX</strong></td>
<td><strong>STAT_TX[1:0]</strong></td>
<td><strong>EA[3:0]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rc_w0</td>
<td>t</td>
<td>t</td>
<td>r</td>
<td>nw</td>
<td>nw</td>
<td>rc_w0</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

They are also reset when an USB reset is received from the USB bus or forced through bit FRES in the CTLR register, except the CTR_RX and CTR_TX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint has its USB_EPnR register where \( n \) is the endpoint identifier.

Read-modify-write cycles on these registers should be avoided because between the read and the write operations some bits could be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an 'invariant' value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their 'invariant' value.
Bit 15 **CTR_RX**: Correct transfer for reception

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only ‘0’ can be written, writing 1 has no effect.

Bit 14 **DTOG_RX**: Data toggle, for reception transfers

If the endpoint is not Isochronous, this bit contains the expected value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent to the USB host, following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to Section 39.5.3: Double-buffered endpoints).

If the endpoint is Isochronous, this bit is used only to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to Section 39.5.4: Isochronous transfers). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes ‘0’, the value of DTOG_RX remains unchanged, while writing ‘1’ makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STAT_RX [1:0]**: Status bits, for reception transfers

These bits contain information about the endpoint status, which are listed in Table 249: Reception status encoding on page 1330. These bits can be toggled by software to initialize their value. When the application software writes ‘0’, the value remains unchanged, while writing ‘1’ makes the bit value toggle. Hardware sets the STAT_RX bits to NAK when a correct transfer has occurred (CTR_RX=1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledge a new transaction.

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to Section 39.5.3: Double-buffered endpoints).

If the endpoint is defined as Isochronous, its status can be only ‘VALID’ or ‘DISABLED’, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_RX bits to ‘STALL’ or ‘NAK’ for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

Bit 11 **SETUP**: Setup transaction completed

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (CTR_RX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while CTR_RX bit is at 1; its state changes when CTR_RX is at 0. This bit is read-only.
Bits 10:9 **EP_TYPE[1:0]:** Endpoint type

These bits configure the behavior of this endpoint as described in Table 250: Endpoint type encoding on page 1330. Endpoint 0 must always be a control endpoint and each USB function must have at least one control endpoint which has address 0, but there may be other control endpoints if required. Only control endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK, the USB peripheral will not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint is a control one.

Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EP_KIND configuration bit.

The usage of Isochronous endpoints is explained in Section 39.5.4: Isochronous transfers.

Bit 8 **EP_KIND: Endpoint kind**

The meaning of this bit depends on the endpoint type configured by the EP_TYPE bits. Table 251 summarizes the different meanings.

**DBL_BUF:** This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in Section 39.5.3: Double-buffered endpoints.

**STATUS_OUT:** This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered 'STALL' instead of 'ACK'. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **CTR_TX:** Correct Transfer for transmission

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only '0' can be written.

Bit 6 **DTOG_TX:** Data Toggle, for transmission transfers

If the endpoint is non-Isochronous, this bit contains the required value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to Section 39.5.3: Double-buffered endpoints).

If the endpoint is Isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to Section 39.5.4: Isochronous transfers). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for Isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes '0', the value of DTOG_TX remains unchanged, while writing '1' makes the bit value toggle. This bit is read/write but it can only be toggled by writing 1.
Bits 5:4 **STAT_TX [1:0]:** Status bits, for transmission transfers

These bits contain the information about the endpoint status, listed in Table 252. These bits can be toggled by the software to initialize their value. When the application software writes '0', the value remains unchanged, while writing '1' makes the bit value toggle. Hardware sets the STAT_TX bits to NAK, when a correct transfer has occurred (CTR_TX=1) corresponding to an IN or SETUP (control only) transaction addressed to this endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to Section 39.5.3: Double-buffered endpoints).

If the endpoint is defined as Isochronous, its status can only be "VALID" or "DISABLED". Therefore, the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_TX bits to ‘STALL’ or ‘NAK’ for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing ‘1.

Bits 3:0 **EA[3:0]:** Endpoint address

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

### Table 249. Reception status encoding

<table>
<thead>
<tr>
<th>STAT_RX[1:0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DISABLED: all reception requests addressed to this endpoint are ignored.</td>
</tr>
<tr>
<td>01</td>
<td>STALL: the endpoint is stalled and all reception requests result in a STALL handshake.</td>
</tr>
<tr>
<td>10</td>
<td>NAK: the endpoint is naked and all reception requests result in a NAK handshake.</td>
</tr>
<tr>
<td>11</td>
<td>VALID: this endpoint is enabled for reception.</td>
</tr>
</tbody>
</table>

### Table 250. Endpoint type encoding

<table>
<thead>
<tr>
<th>EP_TYPE[1:0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>BULK</td>
</tr>
<tr>
<td>01</td>
<td>CONTROL</td>
</tr>
<tr>
<td>10</td>
<td>ISO</td>
</tr>
<tr>
<td>11</td>
<td>INTERRUPT</td>
</tr>
</tbody>
</table>

### Table 251. Endpoint kind meaning

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DBL_BUF</td>
</tr>
<tr>
<td>01</td>
<td>STATUS_OUT</td>
</tr>
<tr>
<td>10</td>
<td>Not used</td>
</tr>
<tr>
<td>11</td>
<td>Not used</td>
</tr>
</tbody>
</table>
Table 252. Transmission status encoding

<table>
<thead>
<tr>
<th>STAT_TX[1:0]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DISABLED: all transmission requests addressed to this endpoint are ignored.</td>
</tr>
<tr>
<td>01</td>
<td>STALL: the endpoint is stalled and all transmission requests result in a STALL handshake.</td>
</tr>
<tr>
<td>10</td>
<td>NAK: the endpoint is naked and all transmission requests result in a NAK handshake.</td>
</tr>
<tr>
<td>11</td>
<td>VALID: this endpoint is enabled for transmission.</td>
</tr>
</tbody>
</table>
39.6.2 Buffer descriptor table

Note: The buffer descriptor table is located inside the packet buffer memory in the separate "USB SRAM" address space.

Although the buffer descriptor table is located inside the packet buffer memory ("USB SRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USB SRAM base address. The buffer descriptor table entry associated with the USB_EPnR registers is described below. The packet memory should be accessed only by byte (8-bit) or half-word (16-bit) accesses. Word (32-bit) accesses are not allowed.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in Structure and usage of packet buffers on page 1308.

Transmission buffer address n (USB_ADDRn_TX)

Address offset: [USB_BTABLE] + n*8

Note: In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB_ADDRn_TX_0.

In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB_ADDRn_RX_0.

<table>
<thead>
<tr>
<th>Bits 15:1 ADDRn_TX[15:1]</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw -</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:1 ADDRn_TX[15:1]: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Bit 0 Must always be written as '0 since packet memory is half-word wide and all packet buffers must be half-word aligned.

Transmission byte count n (USB_COUNTn_TX)

Address offset: [USB_BTABLE] + n*8 + 2

Note: In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB_COUNTn_TX_0.

In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB_COUNTn_RX_0.

<table>
<thead>
<tr>
<th>Bits 15:0 COUNTn_TX[9:0]</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw -</td>
<td></td>
</tr>
</tbody>
</table>

1332/1530 RM0434 Rev 13
Bits 15:10 These bits are not used since packet size is limited by USB specifications to 1023 bytes. Their value is not considered by the USB peripheral.

Bits 9:0 COUNTn_TX[9:0]: Transmission byte count
These bits contain the number of bytes to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Reception buffer address n (USB_ADDRn_RX)
Address offset: [USB_BTABLE] + n*8 + 4

Note: In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB_ADDRn_RX_1.
In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB_ADDRn_TX_1.

<table>
<thead>
<tr>
<th>ADDRn_RX[15:1]</th>
<th>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</th>
</tr>
</thead>
</table>

Bits 15:1 ADDRn_RX[15:1]: Reception buffer address
These bits point to the starting address of the packet buffer, which will contain the data received by the endpoint associated with the USB_EPnR register at the next OUT/SETUP token addressed to it.

Bit 0 This bit must always be written as '0' since packet memory is half-word wide and all packet buffers must be half-word aligned.

Reception byte count n (USB_COUNTn_RX)
Address offset: [USB_BTABLE] + n*8 + 6

Note: In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB_COUNTn_RX_1.
In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB_COUNTn_TX_1.

<table>
<thead>
<tr>
<th>BLSIZE</th>
<th>NUM_BLOCK[4:0]</th>
<th>COUNTn_RX[9:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw rw rw rw rw</td>
</tr>
</tbody>
</table>

This table location is used to store two different values, both required during packet reception. The most significant bits contain the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint descriptor and it is normally defined during the
enumeration process according to its maxPacketSize parameter value (See “Universal Serial Bus Specification”).

Bit 15  **BL_SIZE**: Block size
This bit selects the size of memory block used to define the allocated buffer area.
- If BL_SIZE=0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BL_SIZE=1, the memory block is 32-byte large, which allows to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 14:10  **NUM_BLOCK[4:0]**: Number of blocks
These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BL_SIZE value as illustrated in Table 253.

Bits 9:0  **COUNTn_RX[9:0]**: Reception byte count
These bits contain the number of bytes received by the endpoint associated with the USB_EPnR register during the last OUT/SETUP transaction addressed to it.

### Table 253. Definition of allocated buffer memory

<table>
<thead>
<tr>
<th>Value of NUM_BLOCK[4:0]</th>
<th>Memory allocated when BL_SIZE=0</th>
<th>Memory allocated when BL_SIZE=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (‘00000)</td>
<td>Not allowed</td>
<td>32 bytes</td>
</tr>
<tr>
<td>1 (‘00001)</td>
<td>2 bytes</td>
<td>64 bytes</td>
</tr>
<tr>
<td>2 (‘00010)</td>
<td>4 bytes</td>
<td>96 bytes</td>
</tr>
<tr>
<td>3 (‘00011)</td>
<td>6 bytes</td>
<td>128 bytes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>14 (‘01110)</td>
<td>28 bytes</td>
<td>480 bytes</td>
</tr>
<tr>
<td>15 (‘01111)</td>
<td>30 bytes</td>
<td></td>
</tr>
<tr>
<td>16 (‘10000)</td>
<td>32 bytes</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>29 (‘11101)</td>
<td>58 bytes</td>
<td></td>
</tr>
<tr>
<td>30 (‘11110)</td>
<td>60 bytes</td>
<td></td>
</tr>
<tr>
<td>31 (‘11111)</td>
<td>62 bytes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
39.6.3 USB register map

The table below provides the USB register map and reset values.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>0x20</th>
<th>0x4C</th>
<th>0x3F</th>
<th>0x40</th>
<th>0x44</th>
<th>0x48</th>
<th>0x50</th>
<th>0x58</th>
<th>0x60</th>
<th>0x70</th>
<th>0x80</th>
<th>0x90</th>
<th>0x100</th>
<th>0x120</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>USB_EP0R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 0 1 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>0x04</td>
<td>USB_EP1R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x08</td>
<td>USB_EP2R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 1 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>0x0C</td>
<td>USB_EP3R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x10</td>
<td>USB_EP4R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x14</td>
<td>USB_EP5R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x18</td>
<td>USB_EP6R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x1C</td>
<td>USB_EP7R</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x20-</td>
<td></td>
<td></td>
<td></td>
<td></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>0x20</td>
<td>USB_CNTR</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x40</td>
<td>USBISTR</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x48</td>
<td>USB_FNR</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>0x4C</td>
<td>USB_DADDR</td>
<td></td>
<td></td>
<td></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 0 0 0 0 0 0 0 0 0 0 0 0 1 0 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>
</tbody>
</table>
Refer to Section 2.2 on page 66 for the register boundary addresses.
40  Clock recovery system (CRS)

40.1  CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from the LSE oscillator output, or generated by user software.

40.2  CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - LSE oscillator output
  - packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

40.3  CRS implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIM width</td>
<td>6 bits</td>
</tr>
</tbody>
</table>
40.4 CRS functional description

40.4.1 CRS block diagram

![Figure 410. CRS block diagram](image)

40.4.2 Synchronization input

For more information on the CRS synchronization source configuration, refer to Section 40.7.2.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

40.4.3 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a
SYNMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 411. CRS counter behavior

40.4.4 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
– SYNCKOK status indicated
– TRIM value not changed in AUTOTRIM mode

When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
– SYNCKOK status indicated
– TRIM value adjusted by one trimming step in AUTOTRIM mode

When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
– SYNCPWARN status indicated
– TRIM value adjusted by two trimming steps in AUTOTRIM mode

When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
– SYNCRERR or SYMCNMISS status indicated
– TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

40.4.5 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

\[ \text{RELOAD} = \left( \frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) - 1 \]

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

\[ \text{FELIM} = \left( \frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) \times \text{STEP} \% / 100 \% / 2 \]

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to \( \left( \frac{f_{\text{TARGET}}}{f_{\text{SYNC}}} \right) = 48000 \), and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.
Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

40.5 CRS low-power modes

Table 256. Effect of low-power modes on CRS

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. CRS interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.</td>
</tr>
<tr>
<td>Standby</td>
<td>The CRS peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>The CRS peripheral is powered down and must be reinitialized after exiting Shutdown mode.</td>
</tr>
</tbody>
</table>

40.6 CRS interrupts

Table 257. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected synchronization</td>
<td>ESYNCF</td>
<td>ESYNCE</td>
<td>ESYNCC</td>
</tr>
<tr>
<td>Synchronization OK</td>
<td>SYNCOKF</td>
<td>SYNCOKIE</td>
<td>SYNCOKC</td>
</tr>
<tr>
<td>Synchronization warning</td>
<td>SYNCWARNF</td>
<td>SYNCWARNIE</td>
<td>SYNCWARNC</td>
</tr>
<tr>
<td>Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)</td>
<td>ERF</td>
<td>ERRIE</td>
<td>ERRRC</td>
</tr>
</tbody>
</table>
40.7  **CRS registers**

Refer to *Section 1.2 on page 61* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

### 40.7.1  **CRS control register (CRS_CR)**

Address offset: 0x00  
Reset value: 0x0000 2000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:14</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>13:8</td>
<td>TRIM[5:0]: HSI48 oscillator smooth trimming</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The default value is 32, corresponding to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SWSYNC: Generate software SYNC event</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No action</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: A software SYNC event is generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>AUTOTRIMEN: Automatic trimming enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to <em>Section 40.4.4</em> for more details.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Automatic trimming disabled, TRIM bits can be adjusted by the user.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>CEN: Frequency error counter enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit enables the oscillator clock for the frequency error counter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Frequency error counter disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Frequency error counter enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ESYNCFIE: Expected SYNC interrupt enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Expected SYNC (ESYNCF) interrupt disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Expected SYNC (ESYNCF) interrupt enabled</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**40.7.2 CRS configuration register (CRS_CFRG)**

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>SYNCPOL</th>
<th>SYNCSRC[1:0]</th>
<th>SYNCDIV[2:0]</th>
<th>FELIM[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>RELOAD[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 31 SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.

- 0: SYNC active on rising edge (default)
- 1: SYNC active on falling edge

**Bit 30** Reserved, must be kept at reset value.

**Bits 29:28 SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source.

- 00: GPIO selected as SYNC signal source
- 01: LSE selected as SYNC signal source
- 10: USB SOF selected as SYNC signal source (default).
- 11: Reserved

*Note*: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE or reference clock on the GPIOs should be used as SYNC signal.

**Bit 27** Reserved, must be kept at reset value.
Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

- 000: SYNC not divided (default)
- 001: SYNC divided by 2
- 010: SYNC divided by 4
- 011: SYNC divided by 8
- 100: SYNC divided by 16
- 101: SYNC divided by 32
- 110: SYNC divided by 64
- 111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to Section 40.4.4 for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to Section 40.4.3 for more details about counter behavior.

### 40.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08
Reset value: 0x0i00 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>
| FECAP[15:0]: Frequency error capture
| FEDIR: Frequency error direction
| Bit 15 **FEDIR**: Frequency error direction
| 0: Up-counting direction, the actual frequency is above the target
| 1: Down-counting direction, the actual frequency is below the target

<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>
| FEDECAP[15:0]: Frequency error capture
| FEDIR: Frequency error direction
| Bit 15 **FEDIR**: Frequency error direction
| 0: Up-counting direction, the actual frequency is above the target
| 1: Down-counting direction, the actual frequency is below the target

<table>
<thead>
<tr>
<th>14</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>
</tr>
</tbody>
</table>
| TRIMOVF: Trimming overflow or underflow
| Bit 10 **TRIMOVF**: Trimming overflow or underflow
| This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
| 0: No trimming error signaled
| 1: Trimming error signaled

---

1344/1530 RM0434 Rev 13
Bit 9 **SYNCMISS**: SYNC missed
This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
0: No SYNC missed error signaled
1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error
This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.
0: No SYNC error signaled
1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag
This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCF bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCF bit in the CRS_ICR register.
0: No expected SYNC signaled
1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag
This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.
0: No synchronization or trimming error signaled
1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag
This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNR bit in the CRS_ICR register.
0: No SYNC warning signaled
1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag
This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.
0: No SYNC event OK signaled
1: SYNC event OK signaled
40.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C
Reset value: 0x0000 0000

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

Bit 3 ESYNCC: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 ERRC: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 SYNCWARNC: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 SYNCOKC: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

40.7.5 CRS register map

Table 258. CRS register map and reset values
Refer to *Section 2.2 on page 66* for the register boundary addresses.
41 Debug support (DBG)

41.1 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Independent breakpoint debugging of each CPU core in the system
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components.

General features:
- SWJ-DP: JTAG/Serial-wire debug port
- AHB-AP: AHB access port

CPU2 debug features:
- ROM tables
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watchpoint and Trace unit (DWT)
- Cross trigger interface (CTI)

CPU1 debug features
- ROM table
- System control space (SCS)
- Breakpoint unit (FPB)
- Data watchpoint and Trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace macrocell (ETM), only available on STM32WB55xx devices
- Cross trigger interface (CTI)
- Trace port interface unit (TPU)

CPU2 debug access via CPU2 AHB-AP and its associated AHB bus is disabled.

The CPU1 debug features are accessible by the debugger via the CPU1 AHB-AP.

Additional information can be found in the Arm® documents referenced in Section 41.20.

41.2 Debug use cases

The trace and debug system is designed to support a variety of typical use cases:

- Low cost trace
- Limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt
detection and program counter sampling. Single-wire trace can be maintained even when one or both processors are switched off or clock-stopped.

Breakpoint debugging of each core independently
Both processor cores can be simultaneously and independently debugged using equipment connected to the JTAG/SWD debug port. This enables, among others, breakpoint and watchpoint setting, code stepping and memory access.

Synchronous debugging of both cores
When one core stops due to a breakpoint or a debugger stop command, the other core can be stopped as well. Similarly, the cores can be restarted at the same time. This allows the user to debug loosely coupled applications, which require the processors to remain synchronized.

Tracing code execution via the trace port
Trace information from the CPU1 (Cortex®-M4) is combined into a single trace stream and sent to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.
### 41.3 DBG functional description

#### 41.3.1 DBG block diagram

Figure 412. Block diagram of debug support infrastructure

1. Arm® CoreSight™ component

#### 41.3.2 DBG pins and internal signals

Table 259. JTAG/Serial-wire debug port pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>JTAG debug port</th>
<th>SW debug port</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTMS/SWDIO</td>
<td>I JTAG test mode select</td>
<td>IO Serial wire data in/out</td>
</tr>
<tr>
<td>JTCK/SWCLK</td>
<td>I JTAG test clock</td>
<td>I Serial wire clock</td>
</tr>
<tr>
<td>JTDI</td>
<td>I JTAG test data input</td>
<td>-</td>
</tr>
<tr>
<td>JTD0/TRACESWO</td>
<td>O JTAG test data output</td>
<td>-</td>
</tr>
<tr>
<td>nJTRST</td>
<td>I JTAG test reset</td>
<td>-</td>
</tr>
</tbody>
</table>

1. MS4448SV1
41.3.3 DBG power domains

The debug components are located in the core power domain. This means that debugger connection is not possible in shutdown or standby low power modes. To avoid losing the connection when the device enters standby mode, it is possible to maintain the power to the core by setting a bit in the DBGMCU. This keeps the processor clocks active, and holds off the reset, so that the debug session is maintained.

41.3.4 DBG clocks

The debugger supplies the clock for the debug port via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DAPCLK domain, which covers the rest of the SWJ-DP and the CPU2 access port.

The DAPCLK is a gated version of the system HCLK4.

The DAPCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the debug port CTRL/STAT register. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in the debug port CTRL/STAT register. The DAPCLK is disabled at power up, after OBL, and after wakeup from Standby, and must be disabled when the debugger is disconnected, to reduce power consumption.
The debug and trace components included in the processors (among them ETM, ITM, DWG, FPB) are clocked with the corresponding core clock.

41.3.5 Debug and low power modes

The devices include power saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off, or the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power saving mode emulation has been implemented. If emulation is enabled for a domain, the domain still enters power saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode, but the debugger does not lose the connection.

Emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information refer to Section 41.8.

41.3.6 DBG reset

The debug port (SWJ-DP) is reset by a power-on reset or an OBL reset, and when waking up from Standby mode.

41.4 Serial wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a Coresight™ component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:
   – a 5-pin standard JTAG interface (JTAG-DP)
   – a 2-pin (clock + data) “serial-wire debug” port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default the JTAG-DP is selected after a system or a power-on reset. The five IO pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

   ... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions. It should be noted that all SWJ port IOs can be reconfigured to other functions by software, but debugging is no longer be possible.

41.4.1 JTAG debug port

There are two TAPs on the JTAG debug port, the JTAG-DP TAP and the BSC TAP.
The JTAG-DP implements a TAP state machine (TAPSM), shown in Figure 413, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR), and the other one with a number of data registers (DR).

**Figure 413. JTAG TAP state machine**

The operation of the JTAG-DP is as follows:

When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the Instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.

While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
- The LSB of the IR scan chain is output on JTDO.
- Bit[n] of the IR scan chain is transferred to bit[n-1].
- The value on JTDI is transferred to the MSB of the IR scan chain.

When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the Instruction register.

When the TAPSM goes through the Capture-DR state, a value is transferred from one
of the Data registers onto one of the DR scan chains, connected between JTDI and JTDO.

The value held in the Instruction register determines which Data register (and associated DR scan chain) is selected.

This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.

When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected Data register.

When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in Table 263. The total IR instruction length is 9 bits.

<table>
<thead>
<tr>
<th>IR instruction</th>
<th>DR register</th>
<th>Scan chain length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 to 0111</td>
<td>(BYPASS)</td>
<td>1</td>
<td>Not implemented: BYPASS selected</td>
</tr>
</tbody>
</table>
| 1000           | ABORT       | 35                | ABORT register  
|-- Bits 31:1 = Reserved  
|-- Bit 0 = APABORT: write 1 to generate an AP abort. |
| 1001           | (BYPASS)    | 1                 | Reserved: BYPASS selected |
| 1010           | DPACC       | 35                | Debug port access register  
|                |             |                   | Initiates the debug port and gives access to a debug port register.  
|                |             |                   | -- When transferring data IN:  
|                |             |                   | Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request  
|                |             |                   | Bit 0 = RnW = Read request (1) or write request (0).  
|                |             |                   | -- When transferring data OUT:  
|                |             |                   | Bits 34:3 = DATA[31:0] = 32-bit data read following a read request  
|                |             |                   | Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:  
|                |             |                   | 010 = OK/FAULT  
|                |             |                   | 001 = WAIT  
|                |             |                   | OTHER = reserved |
| 1011           | APACC       | 35                | Access port access register  
|                |             |                   | Initiates an access port and gives access to an access port register.  
|                |             |                   | -- When transferring data IN:  
|                |             |                   | Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request  
|                |             |                   | Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register.  
|                |             |                   | Bit 0 = RnW = Read request (1) or write request (0).  
|                |             |                   | -- When transferring data OUT:  
|                |             |                   | Bits 34:3 = DATA[31:0] = 32-bit data read following a read request  
|                |             |                   | Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:  
|                |             |                   | 010 = OK/FAULT  
|                |             |                   | 001 = WAIT  
|                |             |                   | OTHER = reserved |
| 1100           | (BYPASS)    | 1                 | Reserved: BYPASS selected |
The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification [1].

41.4.2 SW debug port

The Serial Wire Debug protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three phases:

1. packet request (8 bits) transmitted by the host, see Table 264.
2. acknowledge response (3 bits) transmitted by the target, see Table 265.
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read), see Table 266.

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 263. JTAG-DP data registers (continued)

<table>
<thead>
<tr>
<th>IR instruction</th>
<th>DR register</th>
<th>Scan chain length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>(BYPASS)</td>
<td>1</td>
<td>Reserved: BYPASS selected</td>
</tr>
<tr>
<td>1110</td>
<td>IDCODE</td>
<td>32</td>
<td>ID Code 0x6BA0 0477: Arm® JTAG debug port ID code</td>
</tr>
<tr>
<td>1111</td>
<td>BYPASS</td>
<td>1</td>
<td>Bypass A single JTCK cycle delay is inserted between JTDI and JTDI</td>
</tr>
</tbody>
</table>

Table 264. Packet request

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start</td>
<td>Must be “1”</td>
</tr>
</tbody>
</table>
| 1         | APnDP    | – 0: DP register access - see Table 263 for a list of DP registers
|           |          | – 1: AP register access - see Section 41.5: Access ports |
| 2         | RnW      | – 0: Write request
|           |          | – 1: Read request |
| 4:3       | A(3:2)   | Address field of the DP or AP register (refer to ) |
| 5         | Parity   | Single bit parity of preceding bits |
| 6         | Stop     | 0 |
| 7         | Park     | Not driven by host, must be read as “1” by target |
In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial Wire debug protocol, refer to the Arm® Debug Interface Architecture Specification [1].

Note: The SWJ-DP implements SWD protocol version 2.

### 41.4.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in Table 267.

The debugger can access the DP registers as follows:

- Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a Read or Write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit reset (see Table 264). The write data are sent in the data phase.

- To access one of the banked DP registers at address 0x4, the register number must first be written to the DP_SELECT register at address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the content of the DP_SELECT register.

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:0</td>
<td>ACK</td>
<td>– 000: 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>

Table 265. ACK response

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>WDATA or RDATA</td>
<td>Write or Read data</td>
</tr>
<tr>
<td>32</td>
<td>Parity</td>
<td>Single bit parity of 32 data bits</td>
</tr>
</tbody>
</table>

Table 266. Data transfer
41.4.4 DP debug port identification register (DP_DPIDR)

Address offset: 0x0
Reset value: 0x5BA0 2477
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>
</table>
| Bits 31:28 **REVISION[3:0]**: Revision code
| 0x5 |
| Bits 27:20 **PARTNO[7:0]**: Part number for the debug port
| 0xBA |
| Bits 19:17 Reserved, must be kept at reset value. |
| Bit 16 **MIN**: Minimal debug port (MINDP) implementation
| 0x0: MINDP not implemented (transaction counter and pushed operations are supported)
| Bits 15:12 **VERSION[3:0]**: DP architecture version
| 0x2: DPv2 |
| Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
| 0x23B: Arm® JEDEC code |
| Bit 0 Reserved, must be kept at reset value. |

41.4.5 DP abort register (DP_ABORTR)

Address offset: 0x0
Reset value: 0x0000 0000
Write 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>
</table>
| Bits 31:28 **REVISION[3:0]**: Revision code
| 0x5 |
| Bits 27:20 **PARTNO[7:0]**: Part number for the debug port
| 0xBA |
| Bits 19:17 Reserved, must be kept at reset value. |
| Bit 16 **MIN**: Minimal debug port (MINDP) implementation
| 0x0: MINDP not implemented (transaction counter and pushed operations are supported)
| Bits 15:12 **VERSION[3:0]**: DP architecture version
| 0x2: DPv2 |
| Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
| 0x23B: Arm® JEDEC code |
| Bit 0 Reserved, must be kept at reset value. |
Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: Overrun error clear
0: No effect
1: Clear CTRL/STAT.STICKYORUN bit

Bit 3 **WDERRCLR**: Write data error clear
0: No effect
1: Clear CTRL/STAT.WDATAERR bit

Bit 2 **STKERRCLR**: Sticky error clear
0: No effect
1: Clear CTRL/STAT.STICKYERR bit

Bit 1 **STKCOMPCLR**: Sticky compare clear
0: No effect
1: Clear CTRL/STAT.STICKYCMP bit

Bit 0 **DAPABORT**: Aborts current AP transaction if an excessive number of WAIT responses are returned, indicating that the transaction is stalled.
0: No effect
1: Abort transaction

### 41.4.6 DP control and status register (DP_CTRL/STATR)

Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 0

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>TRNCTL[11:4]</td>
<td></td>
<td></td>
<td></td>
<td></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 14 13 12 11 10 9 8 7 6 5 4 3 2 1 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>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 29 **CDBGPWRUPACK**: See description in Section 41.3.4: DBG clocks.
0 = DAPCLK gated
1 = DAPCLK enabled

Bit 28 **CDBGPWRUPREQ**: Controls the DAPCLK enable request signal.
0 = Requests DAPCLK gating
1 = Requests DAPCLK enable

Bits 27:24 Reserved, must be kept at reset value.
Bits 23:12 **TRNCNT[11:0]**: Transaction counter. To program a sequence of transactions to incremental addresses via an AP, TRNCNT is loaded with the number of transactions to perform. It is decremented at the successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]**: Indicates the bytes to be masked in pushed-compare and pushed-verify operations (CTRL/STAT.TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.

- 0b1XXX = include byte lane 3 in comparisons
- 0bX1XX = include byte lane 2 in comparisons
- 0bXX1X = include byte lane 1 in comparisons
- 0bXXX1 = include byte lane 0 in comparisons

Bit 7 **WDATAERR**: Write data error (read only) in SW-DP. Indicates that:
- there is a parity or framing error on the data phase of a write, or
- a write that has been accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the ABORT.WDERRCLR bit.

- 0: No error
- 1: Error has occurred

Reserved in JTAG-DP.

Bit 6 **READOK**: AP read response (read only) in SW-DP. Indicates the response to the last AP read access.

- 0: Read not OK
- 1: Read OK

Reserved in JTAG-DP.

Bit 5 **STICKYERR**: Transaction error (read only in SW-DP, R/W in JTAG-DP). Indicates that an error occurred in an AP transaction.

- 0: No error
- 1: Error has occurred

In the SW-DP, this bit is reset by writing 1 to the ABORT.STKERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 4 **STICKYCMP**: Compares match (read only in SW-DP, R/W in JTAG-DP). Indicates that a match occurred in a pushed operation.

- 0: Match if TRNMODE = 0x1; no match if TRNMODE = 0x2
- 1: No match if TRNMODE = 0x1; match if TRNMODE = 0x2

In the SW-DP, this bit is reset by writing 1 to the ABORT.STKCMPCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bits 3:2 **TRNMODE[1:0]**: Transfer mode for AP write operations (for read operations, this field must be set to 0x0).

- 0x0: Normal operation. AP transactions are passed directly to the AP.
- 0x1: Pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they do not match, the STICKYCMP bit is set.
- 0x2: Pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they match, the STICKYCMP bit is set.
- 0x3: reserved

In pushed operation, only the data bytes indicated by the MASKLANE field are included in the compare.
Bit 1 **STICKYORUN**. Overrun (read only in SW-DP, R/W in JTAG-DP). Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.
- 0: No overrun
- 1: Overrun occurred

In the SW-DP, this bit is reset by writing 1 to the ABORT.ORUNERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0 **ORUNDETECT**. Overrun detection mode enable.
- 0: Overrun detection disabled
- 1: Overrun detection enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions is blocked until the STICKYORUN bit is cleared.

### 41.4.7 DP data link control register (DP_DLCR)
Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 1
Reset value: 0x0000 0040

<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.

Bits 9:8 **TURNROUND[1:0]**: Tristate period for SWDIO.
- 0x0: 1 data bit period
- 0x1: 2 data bit periods
- 0x2: 3 data bit periods
- 0x3: 4 data bit periods

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value (set to 1).

Bits 5:0 Reserved, must be kept at reset value.

### 41.4.8 DP target identification register (DP_TARGETIDR)
Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 2
Reset value: 0x0495 0041

<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>
</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>TPARTNO[3:0]</td>
<td>TDESIGNER[10:0]</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>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 31:28 TREVISION: Target revision.
0x0: revision 1

Bits 27:12 TPARTNO: Target part number.
0x4950: STM32WB55xx/35xx

Bits 11:1 TDESIGNER: Target designer JEDEC code.
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value (set to 1)

41.4.9 DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4 and DP_SELECTR.DPBANKSEL = 3
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>

<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 TINSTANCE[3:0]: Target instance number. Defines the instance number for this device in a multi-drop system.
0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 PROTSVN[3:0]: Serial Wire Debug protocol version.
0x1: Version 2

41.4.10 DP resend register (DP_RESENDR)

Address offset: 0x8
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>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>

<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 RESEND: Returns the value that was returned by the last AP read or DP RDBUFF read. Used in the event of a corrupted read transfer.
41.4.11  DP access port select register (DP_SELECTR)

Address offset: 0x8
Reset value: Unknown

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-24</td>
<td>APSEL[7:0]: Access port select. Selects the access port for the next transaction.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0: AP0 - CPU1 (Cortex®-M4) debug access port (AHB-AP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1: AP1 - CPU2 (Cortex®-M0+) debug access port (AHB-AP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2 to 0xFF: reserved</td>
<td></td>
</tr>
<tr>
<td>27-8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>7-4</td>
<td>APBANKSEL[3:0]: AP register bank select. Selects the 4-word register bank on the active AP for the next transaction.</td>
<td></td>
</tr>
<tr>
<td>3-0</td>
<td>DPBANKSEL[3:0]: DP register bank select. Selects the register at address 0x4 of the debug port.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0: CTRL/STAT register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1: DLCR register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x2: TARGETID register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x3: DLPIDR register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x4 to 0xF: Reserved</td>
<td></td>
</tr>
</tbody>
</table>

41.4.12  DP read buffer register (DP_BUFFR)

Address offset: 0xC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>RDBUFF[31:16]: Contains the value that was returned by the last AP read access. The value returned by an AP read access can either be obtained using a second read access to the same address, which initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.</td>
<td></td>
</tr>
<tr>
<td>15-0</td>
<td>RDBUFF[15:0]:</td>
<td></td>
</tr>
</tbody>
</table>
### 41.4.13 DP target selection register (DP_TARGETSELR)

Address offset: 0xC
Reset value: Unknown

<table>
<thead>
<tr>
<th>TINSTANCE[3:0]</th>
<th><strong>TPARTNO[15:4]</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>w w w w</td>
<td>w w w w w w w w w w</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TPARTNO[3:0]</th>
<th>TDESIGNER[10:0]</th>
<th><strong>Res.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>w w w w</td>
<td>w w w w w w w w w w</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 **TINSTANCE[3:0]**: Target instance number. Defines the instance number for the target device in a multi-drop system. These bits must be written with the same value used for DLPIDR.TINSTANCE to select this device.

Bits 27:12 **TPARTNO[15:0]**: Target part number. Defines the part number for the target device. These bits must be written with the same value used for TARGETID.TPARTNO to select this device.

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code. Defines the JEDEC code for the target device. These bits must be written with the same value used for TARGETID.TDESIGNER to select this device.

Bit 0 Reserved, must be kept at reset value (set to 1).
### 41.4.14 Debug port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address is 2-bit wide, defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

#### Table 267. Debug port register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>0b00</th>
<th>0b01(1)</th>
<th>0b01(2)</th>
<th>0b01(3)</th>
<th>0b10</th>
<th>0b11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DP_DPIDR</td>
<td>24</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>REVISION [3:0]</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PARTNO[7:0]</td>
<td>26</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>MIN</td>
<td>18</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>VERSION [3:0]</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DESIGNER[10:0]</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_ABORTR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_CTRL/STATR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_DLCR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_TARGETIDR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TREVISION [3:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TPARTNO[15:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TDESIGNER[10:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_DLPIDR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TINSTANCE [3:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PROTSVN [3:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_RESENR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RESEND[0.31]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_SELECTR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>APSEL[7:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>APBANKSEL [3:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DPBANKSEL [3:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_BUFFR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RDBUFF[0.31]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DP_TERGETSELR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TINSTANCE [3:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TPARTNO[15:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TDESIGNER[10:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1. DP_SELECTR.DPBANKSEL = 1.
2. DP_SELECTR.DPBANKSEL = 2.
3. DP_SELECTR.DPBANKSEL = 3.
41.5 Access ports

As shown in Figure 414, there are two access ports (AP) attached to the DP:

1. AP0: CPU1 (Cortex®-M4) access port (AHB-AP): enables access to the debug and trace features integrated in the Cortex®-M4 processor core via its internal AHB bus.
2. AP1: CPU2 (Cortex®-M0+) access port (AHB-AP): enables access to the debug and trace features integrated in the Cortex®-M0+ processor core via its internal AHB bus.

Both access ports are of type MEM-AP, the debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in Table 268.

Figure 414. Debug and access port connections

The address of the AP registers is composed of:

- Bits [7:4]: content of the SELECT register APBANKSEL field in the DP (see Section 41.4.11)
- Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see Table 267) or of the SW-DP Packet Request (see Table 264), depending on the debug interface used
- Bits [1:0]: Always set to 0

The content of the SELECT register APSEL field in the DP defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the SELECT register APSEL field in the DP to choose one of the APs, and the APBANKSEL field to select the register bank to be accessed (see Section 41.4.11).
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a Read or Write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit set (see Table 264). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the MEM-AP registers (i.e. using the above AP register access procedure) as follows:
1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to Banked data register BDn triggers an access to address TAR[31:4] + n (this enables up to four consecutive addresses to be accessed without changing the address in the TAR register).

Figure 415 shows how the MEM-AP is used to connect the debug port to the debug components (in this example, a processor, an ETM and a ROM table).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification [1].
Figure 415. Debugger connection to debug components

This diagram illustrates the connection of the debugger to debug components. The figure shows how data and addresses are transferred between the debugger and the debug components. The diagram includes various registers and ports, such as APACC and DPACC, which are used for controlling and accessing debug information. The text explains that this is a partial view of the DP registers and provides more information on the DP descriptions.
### 41.5.1 AP control/status word register (AP_CSWR)

Address offset: 0x0

Reset value: 0x2300 0040

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30 | **SPROT**: Secure transfer request. In the AHB-APs this field sets the protection attribute HPROT[6] of the bus transfer. |
|        | 0: If SPIDEN is high, secure transfer. If SPIDEN is low, non-secure transfer. |
|        | 1: Non-secure transfer. |
| Bit 29 | Reserved, must be kept at reset value. |
| Bits 28:24 | **PROT[4:0]**: Bus transfer protection. In the AHB-APs this field sets the protection attributes HPROT[4:0] of the bus transfer. |
|          | 0bXXXX0: Instruction fetch |
|          | 0bXXXX1: Data access |
|          | 0bXXXXX: User mode |
|          | 0bXXX1X: Privileged mode |
|          | 0bX0XX: Non-bufferable |
|          | 0bX1XX: Bufferable |
|          | 0b0XXX: Non-cacheable |
|          | 0b1XXX: Cacheable |
|          | 0b0XX: Non-exclusive |
|          | 0b1XX: Exclusive |
| Bit 23 | **SPISTATUS**: Status of SPIDEN option bit (read only). This signal determines whether the debugger can access secure memory. |
|        | 0: Secure AHB transfers are blocked |
|        | 1: Secure AHB transfers are allowed |
| Bits 22:12 | Reserved, must be kept at reset value. |
| Bits 11:8 | **MODE[3:0]**: Barrier support enabled. Defines if memory barrier operation is supported. |
|        | 0x0: Not supported |
| Bit 7 | **TRINPROG**: Transfer in progress (read only). Indicates if a bus transfer is in progress on the AP. |
|        | 0x0: No transfer in progress. |
|        | 0x1: Bus transfer in progress. |
| Bit 6 | **DEVICEEN**: Device enabled (read only). Defines whether the AP can be accessed. |
|        | 0x1: AP access enabled. |
Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode. Defines whether TAR address is automatically incremented after a transaction.
- 0x0: No auto-increment
- 0x1: Address is incremented by the size in bytes of the transaction (SIZE field).
- 0x2: Packed transfers enabled. A 32-bit AP access gives rise to 1 x 32-bit, 2 x 16-bit, or 4 x 8-bit bus transactions, corresponding to the programmed transaction size. Data are packed or unpacked accordingly.
- 0x3: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction
- 0x0: Byte (8-bit)
- 0x1: Halfword (16-bit)
- 0x2: Word (32-bit)
- 0x3-0x7: reserved

### 41.5.2 AP transfer address register (AP_TAR)

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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 **TA[31:0]**: Address of current transfer

### 41.5.3 AP data read/write register (AP_DRWR)

Address offset: 0x0C
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 **TD[31:0]**: Data of current transfer

### 41.5.4 AP banked data registers (AP_BD0-3R)

Address offset: 0x10 (DB0R)
Address offset: 0x14 (BD1R)
Address offset: 0x18 (BD2R)
Address offset: 0x1C (BD3R)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>TBD[31:16]</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>rw</td>
<td>30</td>
<td>rw</td>
<td>29</td>
<td>rw</td>
<td>28</td>
<td>rw</td>
<td>27</td>
<td>rw</td>
<td>26</td>
<td>rw</td>
<td>25</td>
<td>rw</td>
<td>24</td>
<td>rw</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
<td>21</td>
<td>rw</td>
<td>20</td>
<td>rw</td>
<td>19</td>
<td>rw</td>
<td>18</td>
<td>rw</td>
<td>17</td>
<td>rw</td>
<td>16</td>
<td>rw</td>
<td>15</td>
<td>rw</td>
<td>14</td>
</tr>
</tbody>
</table>

Bits 31:0 **TBD[31:0]**: Banked data of current transfer to address TAR. TA+ AP_BDnR address [3:2] + 0b00. Auto address incrementing is not performed on AP_BD0-3R. Banked transfers are only supported for word transfers.

### 41.5.5 AP base address register (AP_BASER)

Address offset: 0xF8
Reset value: 0xE00FF F003 (AP0), 0xF000 0003 (AP1)

<table>
<thead>
<tr>
<th></th>
<th>BASEADDR[19:4]</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>r</td>
<td>30</td>
<td>r</td>
<td>29</td>
<td>r</td>
<td>28</td>
<td>r</td>
<td>27</td>
<td>r</td>
<td>26</td>
<td>r</td>
<td>25</td>
<td>r</td>
<td>24</td>
<td>r</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>r</td>
<td>21</td>
<td>r</td>
<td>20</td>
<td>r</td>
<td>19</td>
<td>r</td>
<td>18</td>
<td>r</td>
<td>17</td>
<td>r</td>
<td>16</td>
<td>r</td>
<td>15</td>
<td>r</td>
<td>14</td>
</tr>
</tbody>
</table>

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of ROM table for the AP. The twelve LSBs are 0 since the ROM table must be aligned on a 4-Kbyte boundary.

- AP0 CPU1 (Cortex®-M4) AHB-AP: 0xE00FF
- AP1 CPU2 (Cortex®-M0+) AHB-AP: 0xF0000

Bits 11:2 Reserved, must be kept at reset value.

- Bit 1 **FORMAT**: Base address register format.
  1: Arm® debug interface v5

- Bit 0 **ENTRYPRESENT**: Indicates that debug components are present on the access port bus.
  1: Debug components are present

### 41.5.6 AP identification register (AP_IDR)

Address offset: 0xFC
Reset value: 0x2477 0011 (AP0), 0x6477 0001 (AP1)
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>
</tbody>
</table>

Bits 31:28 **REVISION[3:0]**:
- 0x2: r0p3
- 0x6: r0p7

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank.
- 0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code.
- 0x3B: Arm®

Bit 16 **MEMAP**: Memory access port.
- 0x1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: Identifies the type of AP.
- 0x11: CPU1 (Cortex®-M4) AHB-AP (AP0)
- 0x01: CPU2 (Cortex®-M0+) AHB-AP (AP1)
41.5.7 Access port register map and reset values

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.


Table 268. Access port register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>AP_CSWR</td>
<td></td>
<td>0x04</td>
<td>AP_TAR</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>AP_DRWR</td>
<td></td>
<td>0x10</td>
<td>AP_BD0R</td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>AP_BD1R</td>
<td></td>
<td>0x18</td>
<td>AP_BD2R</td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>AP_BD3R</td>
<td></td>
<td>0xF8</td>
<td>AP_BASER</td>
<td></td>
</tr>
<tr>
<td>0xFC</td>
<td>AP_IDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset value (AP0): 0 0 0 0 0 0 0 0 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 (AP1): 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1
Reset value (AP0): 0 0 1 0 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 1
Reset value (AP1): 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1
41.6 Cross trigger interface (CTI) and matrix (CTM)

The Cross trigger interfaces (CTIs) and Cross trigger matrix (CTM), taken together, form the CoreSight™ embedded cross trigger (see Figure 416).

There are two CTI components, one dedicated to the CPU2 and one dedicated to the CPU1. The CTIs are connected to each other via the CTM. The CTI registers are accessible to the debugger via the corresponding access port and associated AHB.

The CTIs enable events from various sources to trigger debug and/or trace activity. For example, a breakpoint reached in one of the processor cores can stop the other processor, or a transition detected on an external trigger input can start code trace.

Each CTI has up to eight trigger inputs and eight trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in tables 269 to 272.

Table 269. CPU2 CTI inputs

<table>
<thead>
<tr>
<th>No.</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HALTED</td>
<td>CPU2</td>
<td>CPU2 halted - Indicates CPU2 is in debug mode</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Figure 416. Embedded cross trigger

The CTI registers are accessible to the debugger via the corresponding access port and associated AHB.
Table 269. CPU2 CTI inputs (continued)

<table>
<thead>
<tr>
<th>No.</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Table 270. CPU2 CTI outputs

<table>
<thead>
<tr>
<th>No.</th>
<th>Output signal</th>
<th>Destination component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EDBGREQ</td>
<td>CPU2</td>
<td>CPU2 halt request - Puts CPU2 in debug mode</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>DBGRESTART</td>
<td>CPU2</td>
<td>CPU2 restart request - CPU2 exits debug mode</td>
</tr>
</tbody>
</table>

Table 271. CPU1 CTI inputs

<table>
<thead>
<tr>
<th>No.</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HALTED</td>
<td>CPU1</td>
<td>CPU1 halted - Indicates CPU1 is in debug mode</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>ETMTRIGGER0</td>
<td>CPU1 DWT</td>
<td>Trace trigger - Enables CPU1 execution trace</td>
</tr>
<tr>
<td>5</td>
<td>ETMTRIGGER1</td>
<td>CPU1 DWT</td>
<td>Trace trigger - Enables CPU1 execution trace</td>
</tr>
<tr>
<td>6</td>
<td>ETMTRIGGER2</td>
<td>CPU1 DWT</td>
<td>Trace trigger - Enables CPU1 execution trace</td>
</tr>
<tr>
<td>7</td>
<td>ETMTRIGOUT (1)</td>
<td>CPU1 ETM</td>
<td>ETM triggered - Indicates CPU1 trace active</td>
</tr>
</tbody>
</table>

1. Only available on STM32WB55xx devices.

Table 272. CPU1 CTI outputs

<table>
<thead>
<tr>
<th>No.</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EDBGREQ</td>
<td>CPU1</td>
<td>CPU1 halt request - Puts CPU1 in debug mode</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>
There are four event channels in the cross trigger matrix, thus enabling up to four, parallel, bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number \( m \) on CTI \( x \) to output number \( n \) on CTI \( y \), the input must be connected to an event channel \( p \) using the CTIINEN\( m \) register of CTI \( x \). The same channel \( p \) must be connected to the output using the CTIOUTEN\( n \) register of CTI \( y \).

Note: This applies even if the input and output belong to the same CTI.

An input can be connected to more than one channel (up to four), so an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

### Table 272. CPU1 CTI outputs (continued)

<table>
<thead>
<tr>
<th>No.</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>ETMEXTIN0</td>
<td>CPU1 ETM</td>
<td>ETM trig 0 request - Enables CPU1 execution trace</td>
</tr>
<tr>
<td>5</td>
<td>ETMEXTIN1</td>
<td>CPU1 ETM</td>
<td>ETM trig 0 request - Enables CPU1 execution trace</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>DBGRESTART</td>
<td>CPU1</td>
<td>CPU1 restart request - CPU1 exits debug mode</td>
</tr>
</tbody>
</table>

### Example configurations

When either CPU core hits a breakpoint, stop the other core. Restart the two cores synchronously.

To stop both cores when one of them stops the HALTED output of each core must be connected to the EDBGRQ input of the opposite core.

Referring to Table 269 and Table 271, we see that the HALTED signal from the CPU2 is connected to input 0 of the CPU2 CTI, and the same signal from the CPU1 is connected to the same input on the CPU1 CTI. Hence we program the CTIEN0 register on each CTI to connect these inputs to a CTM channel (eg. channel 0).

From Table 270 and Table 272 we see that the EDBGRQ signals to the CPUs are connected to output 0 of the respective CTIs. So we program the CTIOUTEN0 register on each CTI to connect these outputs to the same CTM channel.
To restart both cores simultaneously the debugger must use the APPPULSE register in one of the CTIs. This allows the debugger to generate a pulse on any of the four CTM channels. The channel must be connected to the DBGRESTART signal of both cores.

From *Table 270* and *Table 272* we see that the DBGRESTART signals to the CPUs are connected to output 1 of the respective CTIs. So we program the CTIOUTEN1 register on each CTI, to connect these outputs to an unused CTM channel (e.g. channel 1).

The above configuration is illustrated in *Figure 418*.

![Figure 418. Cross trigger configuration example](image-url)

To force the processors to restart simultaneously, use the following procedure:

1. Clear the debug request by writing 0x01, then 0x00, to the CTIINTACK register in each CTI.
2. Cause a pulse on channel 1 by writing 0x02 to the APPPULSE register in either CTI. This generates a restart request to both processors.

Note that the debugger can also force both cores to stop simultaneously by writing 0x01 to the APPPULSE register in either CTI, which generates a pulse on channel 0.

For more information on the Cross-Trigger Interface CoreSight™ component refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [2].
41.7 Cross trigger interface registers

The register file base address is 0xE0043000 for CPU1 CTI and 0xF0001000 for CPU2 CTI. CPU1 CTI and CPU2 CTI are accessed via different access ports. The registers are the same for each CTI.

41.7.1 CTI control register (CTI_CONTROLR)

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>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 **GLBEN**: Global enable.

0: Cross-triggering disabled
1: Cross-triggering enabled

41.7.2 CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010
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 31:8 Reserved, must be kept at reset value.

Bit 7:0 **INTACK[7:0]**: Trigger acknowledge.

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

41.7.3 CTI application trigger set register (CTI_APPSETR)

Address offset: 0x014
Reset value: 0x0000 0000
41.7.4 CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018
Reset value: 0x0000 0000

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

Bit 3:0 **APPSET[3:0]**: Set channel event.

Read:
- 0bXXX0: Channel 0 event inactive
- 0bXXX1: Channel 0 event active
- 0bXX0X: Channel 1 event inactive
- 0bXX1X: Channel 1 event active
- 0bX0XX: Channel 2 event inactive
- 0bX1XX: Channel 2 event active
- 0b0XXX: Channel 3 event inactive
- 0b1XXX: Channel 3 event active

Write:
- 0bXXX0: No effect
- 0bXXX1: Set event on Channel 0
- 0bXX0X: No effect
- 0bXX1X: Set event on Channel 1
- 0bX0XX: No effect
- 0bX1XX: Set event on Channel 2
- 0b0XXX: No effect
- 0b1XXX: Set event on Channel 3
Bit 3:0 **APPCLEAR[3:0]**: Clear channel event.
- 0b0000: No effect
- 0bXXX1: Clear event on Channel 0
- 0bXX1X: Clear event on Channel 1
- 0bX1XX: Clear event on Channel 2
- 0b1XXX: Clear event on Channel 3

### 41.7.5 CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Res.</td>
</tr>
<tr>
<td>15-0</td>
<td>Res.</td>
</tr>
</tbody>
</table>

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

Bit 3:0 **APPPULSE[3:0]**: Pulse channel event. This register clears itself immediately.
- 0b0000: No effect
- 0bXXX1: Generate pulse on Channel 0
- 0bXX1X: Generate pulse on Channel 1
- 0bX1XX: Generate pulse on Channel 2
- 0b1XXX: Generate pulse on Channel 3

### 41.7.6 CTI trigger In x enable register (CTI_INENRx)

Address offset: 0x020 + 4 * x, where x = 0 to 7

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Res.</td>
</tr>
<tr>
<td>15-0</td>
<td>Res.</td>
</tr>
</tbody>
</table>

Bits 31:4 Reserved, must be kept at reset value.
### 41.7.7 CTI trigger out x enable register (CTI_OUTENRx)

Address offset: $0x0A0 + 4 * x$, where $x = 0$ to $7$

Reset value: $0x0000 0000$

<table>
<thead>
<tr>
<th>Bit 31:4</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 3:0</td>
<td><strong>TRIGOUTEN[3:0]</strong>: For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).</td>
</tr>
<tr>
<td>0b0000:</td>
<td>Channel events do not generate triggers on Trigger outputs</td>
</tr>
<tr>
<td>0bXXX1:</td>
<td>Channel 0 events generate triggers on Trigger output n</td>
</tr>
<tr>
<td>0bXX1X:</td>
<td>Channel 1 events generate triggers on Trigger output n</td>
</tr>
<tr>
<td>0bX1XX:</td>
<td>Channel 2 events generate triggers on Trigger output n</td>
</tr>
<tr>
<td>0b1XXX:</td>
<td>Channel 3 events generate triggers on Trigger output n</td>
</tr>
</tbody>
</table>

### 41.7.8 CTI trigger in status register (CTI_TRGISTSR)

Address offset: $0x130$

Reset value: $0x0000 0000$

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7:0</td>
<td><strong>TRIGINSTATUS[7:0]</strong>: Trigger input status.</td>
</tr>
<tr>
<td>There is one bit of the register for each CTITRIGIN input. When a bit is set to 1 it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.</td>
<td></td>
</tr>
</tbody>
</table>
41.7.9  CTI trigger out status register (CTI_TRGOSTSR)

Address offset: 0x134
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 31:8  Reserved, must be kept at reset value.

Bit 7:0  **TRIGOUTSTATUS[7:0]**: Trigger output status.

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1 it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

41.7.10 CTI channel in status register (CTI_CHINSTSR)

Address offset: 0x138
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 31:4  Reserved, must be kept at reset value.

Bit 3:0  **CHINSTATUS[3:0]**: Channel input status.

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

41.7.11 CTI channel out status register (CTI_CHOUTSTSR)

Address offset: 0x13C
Reset value: 0x0000 0000
### 41.7.12 CTI channel gate register (CTI_GATER)

Address offset: 0x140  
Reset value: 0x0000 000F

<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 3:0  CHOUTSTATUS[3:0]: Channel output status.

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

### 41.7.13 CTI claim tag set register (CTI_CLAIMSETR)

Address offset: 0xFA0  
Reset value: 0x0000 000F

<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 3:0  GATEEN[3:0]: Channel output enable. For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

- 0b0000: Channels events do not propagate
- 0b0XX1: Channel 0 events propagate
- 0bXX1X: Channel 1 events propagate
- 0bX1XX: Channel 2 events propagate
- 0b1XXX: Channel 3 events propagate

#### Claimset[3:0]  

<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>

1382/1530  RM0434 Rev 13
41.7.14 CTI claim tag clear register (CTICLAIMCLR)

Address offset: 0xFA4

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 CLAIMCLR[3:0]: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

41.7.15 CTI lock access register (CTI_LAR)

Address offset: 0xFB0

Reset value: N/A

<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>ACCESS_W[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></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>
<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>

| ACCESS_W[15:0] |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |
41.7.16 CTI lock status register (CTI_LSR)

Address offset: 0xFB4
Reset value: 0x0000 0003

<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.

- Bit 2 **LOCKTYPE**: Indicates the size of the CTI_LAR register
  - 0: 32-bit
- Bit 1 **LOCKGRANT**: Current status of lock. This bit always reads as zero by an external debugger.
  - 0: Write access is permitted
  - 1: Write access is blocked. Only reads are permitted.
- Bit 0 **LOCKEXIST**: Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.
  - 0: No lock control mechanism exists.
  - 1: Lock control mechanism is implemented

41.7.17 CTI authentication status register (CTI_AUTHSTATR)

Address offset: 0xFB8
Reset value: 0x0000 000A

<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, must be kept at reset value.

- Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
  - 0x0: Not implemented
Bits 5:4  **SID[1:0]**: Security level for secure invasive debug  
0x0: Not implemented  
Bits 3:2  **NSID[1:0]**: Security level for non-secure non-invasive debug  
0x2: Disabled  
0x3: Enabled  
Bits 1:0  **NSID[1:0]**: Security level for non-secure invasive debug  
0x2: Disabled  
0x3: Enabled

### 41.7.18  CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8  
Reset value: 0x0004 0800

<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>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>
<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>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</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>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 31:20  Reserved, must be kept at reset value.  
Bits 19:16  **NUMCH[3:0]**: Number of ECT channels available  
0x4: 4 channels  
Bits 15:8  **NUMTRIG[7:0]**: Number of ECT triggers available  
0x8: 8 trigger inputs and 8 trigger outputs  
Bits 7:5  Reserved, must be kept at reset value.  
Bits 4:0  **ExTMUXNUM[4:0]**: Number of trigger input/output multiplexers  
0x0: None

### 41.7.19  CTI device type identifier register (CTI_DEVTYPER)

Address offset: 0xFCC  
Reset value: 0x0000 0014

<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>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>
<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>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>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</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.
41.7.20  CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<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>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  **4KCOUNT[3:0]**: register file size
  0x0: Register file occupies a single 4 KB region
Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
  0x4: Arm® JEDEC code

41.7.21  CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0006

<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>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  **PARTNUM[7:0]**: Part number bits [7:0]
  0x06: CTI part number

41.7.22  CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B9
Bit 31:8  Reserved, must be kept at reset value.

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

  
  0x9: CTI part number

41.7.23  **CTI CoreSight peripheral identity register 2 (CTI_PIDR2)**

Address offset: 0xFE8

Reset value: 0x0000 004B

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

Bits 7:4  **REVISION[3:0]**: Component revision number
  
  0x4: r0p5

Bit 3  **JEDEC**: JEDEC assigned value
  
  0x1: Designer ID specified by JEDEC

  
  0x3: Arm® JEDEC code

41.7.24  **CTI CoreSight peripheral identity register 3 (CTI_PIDR3)**

Address offset: 0xFEC

Reset value: 0x0000 0000
### 41.7.25 CTI CoreSight component identity register 0 (CTI_CIDR0)

**Address offset:** 0xFF0  
**Reset value:** 0x0000 000D

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bits 7:4 | **REVAND[3:0]**: metal fix version  
|          | 0x0: No metal fix                      |
| Bits 3:0 | **CMOD[3:0]**: Customer modified  
|          | 0x0: No customer modifications        |

**Address offset:** 0xFF0  
**Reset value:** 0x0000 000D

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bits 7:0 | **PREAMBLE[7:0]**: Component ID bits [7:0]  
|          | 0x0D: Common ID value                  |

### 41.7.26 CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

**Address offset:** 0xFF4  
**Reset value:** 0x0000 0090

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

| Bit 31:8 | Reserved, must be kept at reset value. |
Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x9: CoreSight™ component

0x0: Common ID value

### 41.7.27 CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<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 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 41.7.28 CTI CoreSight component identity register 3 (CTI_CIDR3)

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

<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 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value
### CTI register map and reset values

<table>
<thead>
<tr>
<th>Register name</th>
<th>Offset</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTI_CONTROLR</td>
<td>0x000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTI_INTACKR</td>
<td>0x010</td>
<td>INTACK[7:0]</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>CTI_APPSETR</td>
<td>0x014</td>
<td>APPSET[3:0]</td>
<td></td>
</tr>
<tr>
<td>CTI_APPCLEAR</td>
<td>0x018</td>
<td>APPCLEAR[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_APPPULSER</td>
<td>0x01C</td>
<td>APPPULSE[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR0</td>
<td>0x020</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR1</td>
<td>0x024</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR2</td>
<td>0x028</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR3</td>
<td>0x02C</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR4</td>
<td>0x030</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR5</td>
<td>0x034</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR6</td>
<td>0x038</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_INENR7</td>
<td>0x03C</td>
<td>TRIGINEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_OUTENR0</td>
<td>0x040</td>
<td>TRIGOUTEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_OUTENR1</td>
<td>0x054</td>
<td>TRIGOUTEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>CTI_OUTENR2</td>
<td>0x058</td>
<td>TRIGOUTEN[3:0]</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

**Table 273. CTI register map and reset values**
## Table 273. CTI register map and reset values (continued)

| Offset  | Register name   | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value | Reset value |
|---------|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0x05C   | CTI_OUTENR3     | 0 0 0 0     | TRIGOUTEN[3:0] |
| 0x0B0   | CTI_OUTENR4     | 0 0 0 0     | TRIGOUTEN[3:0] |
| 0x0B4   | CTI_OUTENR5     | 0 0 0 0     | TRIGOUTEN[3:0] |
| 0x0B8   | CTI_OUTENR6     | 0 0 0 0     | TRIGOUTEN[3:0] |
| 0x0BC   | CTI_OUTENR7     | 0 0 0 0     | TRIGOUTEN[3:0] |
| 0x130   | CTI_TRIGSTATSR  | 0 0 0 0 0 0 0 0 | TRIGINSTATUS[7:0] |
| 0x134   | CTI_TRIGOSTSR   | 0 0 0 0 0 0 0 0 | TRIGOUTSTATS[7:0] |
| 0x138   | CTI_CHINSTSR    | 0 0 0 0     | CHSTATUS[3:0] |
| 0x13C   | CTI_CHOUTSTATS  | 0 0 0 0     | CHOSTATUS[3:0] |
| 0x140   | CTI_GATER       | 1 1 1 1     | GATE[3:0] |
| 0x1A0   | CTI_CLAIMSETR   | 1 1 1 1     | CLAIMSET[3:0] |
| 0x1A4   | CTI_CLAIMCLR    | 0 0 0 0     | CLAIMCLR[3:0] |
| 0x1B0   | CTI_LAR         | KEY         |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |
| 0x1B4   | CTI_LSR         |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |
| 0x1B8   | CTI_AUTHSTATR   |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |             |
| 0x1C8   | CTI_DEVIDR      | 0 1 0 0 0 0 0 0 0 | NUMCH[3:0] | NUMTRIG[7:0] | EXMUXNUM[4:0] |
| 0x1CC   | CTI_DEVTYPEPER  | 0 0 1 0 1 0 0 | SUB[3:0] | MAJOR[3:0] |             |
Refer to Section 41.9: CPU2 ROM tables and to Section 41.13: CPU1 ROM table for the register boundary addresses.
41.8  Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the debug software) to:

- maintain the clock and power the CPU1 processor core when in low power modes (Sleep, Stop or Standby), CPU2 operation is not influenced in low power debug mode
- maintain the clock and power the system debug and trace components when in low power modes
- stop the clock to certain peripherals (watchdogs, timers, RTC) when either processor core is stopped in debug mode

The DBGMCU registers are not reset by a system reset, only by a power on reset. They are accessible to the debugger via the CPU1 AHB access port at base address 0xE0042000.

Note: The DBGMCU is not a standard CoreSight™ component, consequently it does not appear in the CPU1 ROM table.

41.8.1  DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000
Reset value: 0x200X 6495

<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>t</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</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  \text{REV\_ID}[15:0]: Revision
0x2001 = STM32WB55xx revision Y and STM32WB35xx revision A
0x2003 = STM32WB55xx/35xx revision X

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

Bits 11:0  \text{DEV\_ID}[11:0]: Device ID
0x495: STM32WB55xx/35xx

41.8.2  DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004
Reset value: 0x0000 0000
### 41.8.3 DBGMCU CPU1 APB1 peripheral freeze register 1

**(DBGMCU_APB1FZR1)**

Address offset: 0x03C

Reset value: 0x0000 0000
### DBGMCU CPU2 APB1 peripheral freeze register 1

**DBGMCU_C2APB1FZR1**

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGRQ.

Address offset: 0x040
Reset value: 0x0000 0000

![Table of Freeze Register Bits](image-url)
### 41.8.5 DBGMCU CPU1 APB1 peripheral freeze register 2

**DBGMCU_APB1FZR2**

Address offset: 0x044

Reset value: 0x0000 0000
41.8.6 **DBGMCU CPU2 APB1 peripheral freeze register 2**  
(DBGMCU\_C2APB1FZR2)  

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGRQ.

Address offset: 0x048  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-9</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 5   | **DBG\_LPTIM2\_STOP**: LPTIM2 stop in CPU1 debug  
0: Normal operation. LPTIM2 continues to operate while CPU1 is in debug mode  
1: Stop in debug. LPTIM2 is frozen while CPU1 is in debug mode. |

| Bit 7-0 | Reserved, must be kept at reset value. |

---

41.8.7 **DBGMCU CPU1 APB2 peripheral freeze register**  
(DBGMCU\_APB2FZR)  

Address offset: 0x04C  
Reset value: 0x0000 0000
# Debug support (DBG)

41.8.8 **DBGMCU CPU2 APB2 peripheral freeze register**  
(DBGMCU\_C2APB2FZR)

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGRQ.

Address offset: 0x050  
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 31:19 Reserved, must be kept at reset value.

**Bit 18 DBG\_TIM17\_STOP**  
TIM17 stop in CPU1 debug  
0: Normal operation. TIM17 continues to operate while CPU1 is in debug mode  
1: Stop in debug. TIM17 is frozen while CPU1 is in debug mode.

**Bit 17 DBG\_TIM16\_STOP**  
TIM16 stop in CPU1 debug  
0: Normal operation. TIM16 continues to operate while CPU1 is in debug mode  
1: Stop in debug. TIM16 is frozen while CPU1 is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

**Bit 11 DBG\_TIM1\_STOP**  
TIM1 stop in CPU1 debug  
0: Normal operation. TIM1 continues to operate while CPU1 is in debug mode  
1: Stop in debug. TIM1 is frozen while CPU1 is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

## 41.8.8 DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU\_C2APB2FZR)

CPU2 peripheral freeze is only available when CPU2 debug is enabled or when CPU2 CTI is configured by the debugger for halt via EDBGRQ.

Address offset: 0x050  
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>DBG_TIM1_STOP</td>
<td>DBG_TIM16_STOP</td>
<td>Res.</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 31:19 Reserved, must be kept at reset value.

**Bit 18 DBG\_TIM17\_STOP**  
TIM17 stop in CPU2 debug  
0: Normal operation. TIM17 continues to operate while CPU2 is in debug mode  
1: Stop in debug. TIM17 is frozen while CPU2 is in debug mode.
Bit 17 **DBG_TIM16_STOP**: TIM16 stop in CPU2 debug
- 0: Normal operation. TIM16 continues to operate while CPU2 is in debug mode
- 1: Stop in debug. TIM16 is frozen while CPU2 is in debug mode.

Bits 16:12  Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in CPU2 debug
- 0: Normal operation. TIM1 continues to operate while CPU2 is in debug mode
- 1: Stop in debug. TIM1 is frozen while CPU2 is in debug mode.

Bits 10:0  Reserved, must be kept at reset value.
# Debug MCU register map and reset values

Table 274. DBGMCU 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>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DBGMCU_IDCODE</td>
<td>0x004</td>
<td>DBGMCU_CR</td>
<td>0x008-</td>
<td>Reserved</td>
<td>0x03C</td>
<td>DBGMCU_APB1FZR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x040</td>
<td>DBGMCU_C2APB1FZR1</td>
<td>0x044</td>
<td>DBGMCU_APB1FZR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x048</td>
<td>DBGMCU_C2APB1FZR2</td>
<td>0x04C</td>
<td>DBGMCU_APB2FZR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Register Description

**DBGMCU_IDCODE**

- **Offset:** 0x000
- **Register name:** DBGMCU_IDCODE
- **Functionality:** identifier code
- **Description:**
  - **Reset value:** 0 0 1 0 0 0 0 0 0 0 0 0 X X X X 0 1 1 0 0 1 0 0 1 0 1 0 1

**DBGMCU_CR**

- **Offset:** 0x004
- **Register name:** DBGMCU_CR
- **Functionality:** control register
- **Description:**
  - **Reset value:** 0 0 0 0 0

**DBGMCU_APB1FZR1**

- **Offset:** 0x03C
- **Register name:** DBGMCU_APB1FZR1
- **Functionality:** control register
- **Description:**
  - **Reset value:** 0 0 0 0

**DBGMCU_C2APB1FZR1**

- **Offset:** 0x040
- **Register name:** DBGMCU_C2APB1FZR1
- **Functionality:** control register
- **Description:**
  - **Reset value:** 0 0 0 0

**DBGMCU_APB1FZR2**

- **Offset:** 0x044
- **Register name:** DBGMCU_APB1FZR2
- **Functionality:** control register
- **Description:**
  - **Reset value:** 0

**DBGMCU_C2APB1FZR2**

- **Offset:** 0x048
- **Register name:** DBGMCU_C2APB1FZR2
- **Functionality:** control register
- **Description:**
  - **Reset value:** 0

**DBGMCU_APB2FZR**

- **Offset:** 0x04C
- **Register name:** DBGMCU_APB2FZR
- **Functionality:** control register
- **Description:**
  - **Reset value:** 0 0
Refer to [Section 41.8: Microcontroller debug unit (DBGMCU)](#) 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 |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x050  | DBGMCU      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        | C2APB2FZR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 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  |
41.9 **CPU2 ROM tables**

The ROM tables are CoreSight™ components that contain the base addresses of all the CoreSight™ debug components accessible via the AHBD. These tables allow a debugger to discover the topology of the CoreSight™ system automatically.

There are two ROM tables in the CPU2 sub-system:

1. **ROM1:** CPU2 processor ROM table, pointed to by the BASE register in the CPU2 AHB-AP. It contains the base address pointers for the CTI, as well as for the CPU2 ROM table.

2. **ROM2:** CPU2 ROM table, containing pointers to the CPU2 System control space registers, which allow the debugger to identify the CPU core, as well as to the remaining CoreSight™ components in the CPU2 subsystem (PBU, DWT).

The CPU2 processor ROM table occupies a 4-Kbyte, 32-bit wide chunk of AHB address space, from 0xF0000000 to 0xF0000FFC.

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF0000000</td>
<td>CPU2 ROM table</td>
<td>0xE00FF000</td>
<td>0xF0FF000</td>
<td>4 KB</td>
<td>0xF00FF03</td>
</tr>
<tr>
<td>0xF0000004</td>
<td>CTI</td>
<td>0xF0001000</td>
<td>0x0001000</td>
<td>4 KB</td>
<td>0x00001003</td>
</tr>
<tr>
<td>0xF0000008</td>
<td>Not used</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00002002</td>
</tr>
<tr>
<td>0xF000000C</td>
<td>Not used</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x10000002</td>
</tr>
<tr>
<td>0xF0000010</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xF000000C to 0xF0000FC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xF0000FCC to 0xF000FFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 277</td>
</tr>
</tbody>
</table>

The CPU2 ROM table occupies a 4 KB, 32-bit wide chunk of APB-D address space, from 0xE00FF000 to 0xE00FFFC.

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00FF000</td>
<td>SCS</td>
<td>0xE000E000</td>
<td>0xFFF0F000</td>
<td>4 KB</td>
<td>0xFFF0F003</td>
</tr>
<tr>
<td>0xE00FF004</td>
<td>DWT</td>
<td>0xE001000</td>
<td>0xFFF02000</td>
<td>4 KB</td>
<td>0xFFF02003</td>
</tr>
<tr>
<td>0xE00FF008</td>
<td>BPU</td>
<td>0xE002000</td>
<td>0xFFF03000</td>
<td>4 KB</td>
<td>0xFFF03003</td>
</tr>
<tr>
<td>0xE00FF00C</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FF010 to 0xE00FFFC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FFCC to 0xE00FFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 278</td>
</tr>
</tbody>
</table>
The topology for the CoreSight™ components in the CPU2 subsystem is shown in Figure 419.

**Figure 419. CPU2 CoreSight™ topology**

- **CPU2 Cortex® M0+ Processor ROM table**
  - @0xF0000000
  - Offset: 0x00001000
  - Offset: 0x00002000
  - Offset: 0x10000000

- **CPU2 Cortex® M0+ CPU ROM table**
  - @0xE00FF000
  - Offset: 0xFFF0F000
  - Offset: 0xFFF02000
  - Offset: 0xFFF03000

- **System Control Space (SCS)**
  - @0xE000E000
  - Offset: 0x00001000

- **Register file base**

- **Data Watchpoint/Trace (DWT)**
  - @0xE0001000

- **Breakpoint Unit (BPU)**
  - @0xE0002000

- **Cross Trigger Interface (CTI)**
  - @0xF0000000

- **System Control Space (SCS)**
  - @0xE000E000

- **Register file base**

- **Register file base**

- **Register file base**
### 41.9.1 CPU2 ROM1 memory type register (C2ROM1_MEMTYPE)

Address offset: 0xFCC  
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 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory  
0x1: System memory is present on this bus

### 41.9.2 CPU2 ROM1 CoreSight peripheral identity register 4 (C2ROM1_PIDR4)

Address offset: 0xFD0  
Reset value: 0x0000 0004

<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>4KCOUNT[3:0]</td>
<td>JEP106CON[3:0]</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></td>
<td></td>
<td></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 **4KCOUNT[3:0]**: register file size  
0x0: Register file occupies a single 4 KB region

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

### 41.9.3 CPU2 ROM1 CoreSight peripheral identity register 0 (C2ROM1_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 0C0
### 41.9.4 CPU2 ROM1 CoreSight peripheral identity register 1 (C2ROM1_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B4

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

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
- **0xC0**: Cortex®-M0+ processor ROM table

### 41.9.5 CPU2 ROM1 CoreSight peripheral identity register 2 (C2ROM1_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 00B4

<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>

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

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

- **0x4**: Cortex®-M0+ processor ROM table

### C2ROM1_PIDR2

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

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

- **0x4**: Cortex®-M0+ processor ROM table
Bit 31:8  Reserved, must be kept at reset value.

Bit 7:4  **REVISION[3:0]**: Component revision number
    0x0: rev r0p0

Bit 3  **JEDEC**: JEDEC assigned value
    0x1: Designer ID specified by JEDEC

    0x3: Arm® JEDEC code

### 41.9.6 CPU2 ROM1 CoreSight peripheral identity register 3
(C2ROM1_PIDR3)

Address offset: 0xFEC

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>REVAND[3:0]</td>
<td>CMOD[3:0]</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>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVAND[3:0]**: metal fix version
    0x0: No metal fix

Bits 3:0  **CMOD[3:0]**: Customer modified
    0x0: No customer modifications

### 41.9.7 CPU2 ROM1 CoreSight component identity register 0
(C2ROM1_CIDR0)

Address offset: 0xFF0

Reset value: 0xFF00 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>PREAMBLE[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>
<td></td>
</tr>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</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  **PREAMBLE[7:0]**: Component ID bits [7:0]
    0x0D: Common ID value
41.9.8 CPU2 ROM1 CoreSight peripheral identity register 1
(C2ROM1_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0010

<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>

Bit 31:8 Reserved, must be kept at reset value.
Bits 7:4 CLASS[3:0]: Component ID bits [15:12] - component class
0x01: ROM table component

0x00: Common ID value

41.9.9 CPU2 ROM1 CoreSight component identity register 2
(C2ROM1_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<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, must be kept at reset value.
Bits 7:0 PREAMBLE[19:12]: Component ID bits [23:16]
0x05: Common ID value

41.9.10 CPU2 ROM1 CoreSight component identity register 3
(C2ROM1_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value
### 41.9.11 CPU2 processor ROM table registers and reset values

Table 277. CPU2 processor ROM table register map and reset values

| Offset   | Register name | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|----------|---------------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0FCC   | C2ROM1_MEMTYPE |     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD0    | 4KCOUNT       | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE0    | C2ROM1_PIDR0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE4    | C2ROM1_PIDR1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE8    | C2ROM1_PIDR2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFEC    | C2ROM1_PIDR3  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF0    | C2ROM1_CIDR0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFFFF   | C2ROM1_CIDR1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFFF8   | C2ROM1_CIDR2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFFC    | C2ROM1_CIDR3  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to [Section 41.9: CPU2 ROM tables](#) for the register boundary addresses.
41.9.12  CPU2 ROM2 memory type register (C2ROM2_MEMTYPE)

Address offset: 0xFCC
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>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</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 31:1  Reserved, must be kept at reset value.

Bit 0  **SYSMEM**: System memory
0x1: System memory is present on this bus

41.9.13  CPU2 ROM2 CoreSight peripheral identity register 4 (C2ROM2_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

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

Bits 7:4  **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

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

41.9.14  CPU2 ROM2 CoreSight peripheral identity register 0 (C2ROM2_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0C0

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

Bits 7:4  **4KCOUNT[3:0]**: register file size
0x0: Register file occupies a single 4 KB region

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC continuation code
### 41.9.15 CPU2 ROM2 CoreSight peripheral identity register 1 (C2ROM2_PIDR1)

**Description:**
- **Address offset:** 0xFE4
- **Reset value:** 0x0000 00B4

**Bits:**
- **31:8:** Reserved, must be kept at reset value.
- **7:0:** \texttt{PARTNUM}[7:0]: Part number bits [7:0]
  - 0xC0: CPU2 ROM table

#### Bit 31:8

<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 7:4:** \texttt{JEP106ID}[3:0]: JEP106 identity code bits [3:0]
- 0xB: Arm® JEDEC code

**Bits 3:0:** \texttt{PARTNUM}[11:8]: Part number bits [11:8]
- 0x4: CPU2 ROM table

#### Bit 31:8

<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>

### 41.9.16 CPU2 ROM2 CoreSight peripheral identity register 2 (C2ROM2_PIDR2)

**Description:**
- **Address offset:** 0xFE8
- **Reset value:** 0x0000 00B4

**Bits:**
- **31:8:** Reserved, must be kept at reset value.
- **7:4:** \texttt{JEP106ID}[6:4]: JEP106 identity code bits [6:4]
  - 0xB: Arm® JEDEC code

**Bits 3:0:** \texttt{PARTNUM}[11:8]: Part number bits [11:8]
- 0x4: CPU2 ROM table
Bit 31:8 Reserved, must be kept at reset value.

Bit 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

0x3: Arm® JEDEC code

### 41.9.17 CPU2 ROM2 CoreSight peripheral identity register 3 (C2ROM2_PIDR3)

Address offset: 0xFEC

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>

<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>

**REVAND[3:0]**: metal fix version
0x0: No metal fix

**CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 41.9.18 CPU2 ROM2 CoreSight component identity register 0 (C2ROM2_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

<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>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>

**PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

Bits 31:8 Reserved, must be kept at reset value.
### 41.9.19 CPU2 ROM2 CoreSight peripheral identity register 1
(C2ROM2_CIDR1)

- **Address offset:** 0xFF4
- **Reset value:** 0x0000 0010

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7-4</td>
<td><strong>CLASS[3:0]:</strong> Component ID bits [15:12] - component class</td>
</tr>
<tr>
<td></td>
<td>0x1: ROM table component</td>
</tr>
<tr>
<td>Bits 3-0</td>
<td><strong>PREAMBLE[11:8]:</strong> Component ID bits [11:8]</td>
</tr>
<tr>
<td></td>
<td>0x0: Common ID value</td>
</tr>
</tbody>
</table>

### 41.9.20 CPU2 ROM2 CoreSight component identity register 2
(C2ROM2_CIDR2)

- **Address offset:** 0xFF8
- **Reset value:** 0x0000 0005

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7-0</td>
<td><strong>PREAMBLE[19:12]:</strong> Component ID bits [23:16]</td>
</tr>
<tr>
<td></td>
<td>0x05: Common ID value</td>
</tr>
</tbody>
</table>

### 41.9.21 CPU2 ROM2 CoreSight component identity register 3
(C2ROM2_CIDR3)

- **Address offset:** 0xFFC
- **Reset value:** 0x0000 00B1
### Debug support (DBG)

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

#### Bits 7:0
**PREAMBLE[27:20]:** Component ID bits [31:24]

- 0xB1: Common ID value

<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>
</tbody>
</table>

**PREAMBLE[27:20]:**

- 3r
- r
- r
- r
- r
- r
- r
- r
### 41.9.22 CPU2 ROM table register map and reset values

Table 278. CPU2 ROM table register map and 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>
<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>0xFCC</td>
<td>C2ROM2_MEMTYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFD0</td>
<td>C2ROM2_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFE0</td>
<td>C2ROM2_PIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFE4</td>
<td>C2ROM2_PIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFE8</td>
<td>C2ROM2_PIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFEC</td>
<td>C2ROM2_PIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFF0</td>
<td>C2ROM2_CIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFF4</td>
<td>C2ROM2_CIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFF8</td>
<td>C2ROM2_CIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>0xFFC</td>
<td>C2ROM2_CIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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 41.9: CPU2 ROM tables for the register boundary addresses.
41.10 **CPU2 data watchpoint and trace unit (DWT)**

The DWT provides four comparators that can be used as:

- Watchpoint
- ETM trigger, only available on STM32WB55xx
- PC sampling trigger
- Data address sampling trigger
- Data comparator (for comparator 1 only)
- Clock cycle counter comparator (for comparator 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its DWT_COMPR register with one of the following items:

- a data address
- an instruction address
- a data value
- the cycle count value (comparator 0 only)

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT Data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value.
- A watchpoint debug event, on either the PC value or the accessed data address.
- A CMPMATCH[N] event, that signals the match outside the DWT unit.

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual [5].

### 41.10.1 DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000
Bits 31:28 NUMCOMP: Number of comparators implemented (read only)
  0x4: Four comparators

Bit 27 NOTRCPKT: Trace sampling and exception tracing support (read only)
  0x0: Supported

Bit 26 NOEXTRIG: External match signal, CMPMATCH support (read only)
  0x0: Supported

Bit 25 NOCYCCNT: Cycle counter support (read only)
  0x0: Supported

Bit 24 NOPRFCNT: Profiling counter support (read only)
  0x0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 CYCEVTENA: Enable for POSTCNT underflow event counter packet generation
  0x0: Disabled
  0x1: Enabled

Bit 21 FOLDEVTENA: Enable for folded instruction counter overflow event generation
  0x0: Disabled
  0x1: Enabled

Bit 20 LSUEVTENA: Enable for LSU counter overflow event generation
  0x0: Disabled
  0x1: Enabled

Bit 19 SLEEPEVTENA: Enable for sleep counter overflow event generation
  0x0: Disabled
  0x1: Enabled

Bit 18 EXCEVTENA: Enable for exception overhead counter overflow event generation
  0x0: Disabled
  0x1: Enabled

Bit 17 CPIEVTENA: Enable for CPI counter overflow event generation
  0x0: Disabled
  0x1: Enabled

Bit 16 EXCTRCENA: Enable for exception trace generation
  0x0: Disabled
  0x1: Enabled

Bits 15:13 Reserved, must be kept at reset value.
41.10.2 DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004
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>CYCCNT[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></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>
</tbody>
</table>

Bits 31:0 CYCCNT[31:0]: Processor clock cycle counter

41.10.3 DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008
Reset value: 0x0000 0000
## Debug support (DBG)

### 41.10.4 DWT exception count register (DWT_EXCCNTR)

**Address offset:** 0x00C  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
<th>Bits 7:0</th>
<th><strong>EXCCNT[7:0]</strong></th>
<th>Exception overhead cycle counter. Counts the number of cycles spent in exception processing.</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>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
<td>rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 41.10.5 DWT sleep count register (DWT_SLPCNTR)

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

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
<th>Bits 7:0</th>
<th><strong>SLEEPCNT[7:0]</strong></th>
<th>Sleep cycle counter. Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).</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>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
<td>rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
41.10.6  DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014
Reset value: 0x0000 0000

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  LSUCNT[7:0]: Load store counter. Counts additional cycles required to execute load and store instructions.

41.10.7  DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018
Reset value: 0x0000 0000

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  FOLDCNT[7:0]: Folded instruction counter. Increments on each instruction that takes 0 cycles.

41.10.8  DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C
Reset value: 0x0000 0000
Bits 31:0  **EIASKAMPLE[31:0]**: Executed Instruction Address sample value. Samples the current value of the program counter.

### 41.10.9 DWT comparator register x (DWT_COMPxR)

Address offset: 0x020 + x * 0x10 (for x = 0 to 3)

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>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>f</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**  **COMP[31:0]**: Reference value for comparison.

### 41.10.10 DWT mask register x (DWT_MASKxR)

Address offset: 0x024 + x * 0x10 (for x = 0 to 3)

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 31:0**  **COMP[31:0]**: Reference value for comparison.

Bits 31:5  **Reserved**, must be kept at reset value.

Bits 4:0  **MASK[4:0]**: Comparator mask size. Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

### 41.10.11 DWT function register x (DWT_FUNCTxR)

Address offset: 0x028 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000
Bits 31:25 Reserved, must be kept at reset value.

Bit 24 MATCHED: Comparator match (read only). Indicates if a comparator match has occurred since the register was last read.
0: No match
1: Match occurred

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

Bits 19:16 DATAADDR[3:0]: When the DATAMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 DATAADDR[3:0]: When the DATAMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 DATASIZE[1:0]: For data value matching, specifies the size of the required data comparison.
0x0: Byte
0x1: Half word
0x2: Word
0x3: reserved

Bit 9 LNK1ENA: Indicates whether use of a second linked comparator is supported (read only).
0x1: Supported

Bit 8 DATAMATCH: Enables cycle comparison.
0x0: Perform address comparison
0x1: Perform data value comparison

Bit 7 CYCMATCH: Enables cycle count comparison on comparator 0. This field is reserved for other comparators.
0x0: No cycle count comparison
0x1: Compare DWT_COMP0R with the cycle counter, DWT_CYCCNTR

Bit 6 Reserved, must be kept at reset value.

Bit 5 EMITRANGE: Enables generation of data trace address offset packets (containing data address bits 0 to 15)
0x0: Disabled
0x1: Enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 FUNCTION[3:0]: Selects action to take on comparator match. The meaning of this bit field depends on the setting of the DATAMATCH and CYCMATCH fields. See [5].

41.10.12 DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004
### 41.10.13 DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

<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, must be kept at reset value.

Bits 7:4  **4KCOUNT[3:0]**: register file size
- 0x0: Register file occupies a single 4 KB region

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

<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 Res. Available.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
- 0x02: DWT part number

### 41.10.14 DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

<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>

Bit 31:8 Reserved, must be kept at reset value.
41.10.15 DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 003B

<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>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>

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

Bits 7:4 **REVISION[3:0]**: Component revision number
- 0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
- 0x1: Designer ID specified by JEDEC

- 0x3: Arm® JEDEC code

41.10.16 DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC
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>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>

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

Bits 7:4 **REVAND[3:0]**: metal fix version
- 0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
- 0x0: No customer modifications
### 41.10.17 DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

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

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

### 41.10.18 DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 00E0 

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0xE: Trace generator component

0x0: Common ID value

### 41.10.19 DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8  
Reset value: 0x0000 0005 

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0xE: Trace generator component

0x0: Common ID value
### 41.10.20 DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th></th>
<th>0x05: Common ID value</th>
</tr>
</thead>
</table>

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

Bits 7:0  **PREAMBLE[19:12]**: Component ID bits [23:16]

0x05: Common ID value

<table>
<thead>
<tr>
<th></th>
<th>0xB1: Common ID value</th>
</tr>
</thead>
</table>

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

Bits 7:0  **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value
41.10.21 CPU2 DWT registers

The CPU2 DWT registers are located at address range 0xE0001000 to 0xE0001FFC, on the AHBD.

Table 279. CPU2 DWT 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>0x000</td>
<td>DWT_CTRLR</td>
<td>0x004</td>
<td>DWT_CYCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>DWT_CPICNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>DWT_SLPCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01C</td>
<td>DWT_PCSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>DWT_COMP0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>DWT_MASK0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>DWT_FUNCTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>DWT_COMP1R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x034</td>
<td>DWT_MASK1R</td>
</tr>
</tbody>
</table>

Reset value

<table>
<thead>
<tr>
<th>0x000</th>
<th>DWT_CTRLR</th>
<th>0x004</th>
<th>DWT_CYCCNTR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>DWT_CPICNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>DWT_SLPCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01C</td>
<td>DWT_PCSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>DWT_COMP0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>DWT_MASK0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>DWT_FUNCTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>DWT_COMP1R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x034</td>
<td>DWT_MASK1R</td>
</tr>
</tbody>
</table>

Reset value

<table>
<thead>
<tr>
<th>0x000</th>
<th>DWT_CTRLR</th>
<th>0x004</th>
<th>DWT_CYCCNTR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>DWT_CPICNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>DWT_SLPCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01C</td>
<td>DWT_PCSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>DWT_COMP0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>DWT_MASK0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>DWT_FUNCTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>DWT_COMP1R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x034</td>
<td>DWT_MASK1R</td>
</tr>
</tbody>
</table>

Reset value

<table>
<thead>
<tr>
<th>0x000</th>
<th>DWT_CTRLR</th>
<th>0x004</th>
<th>DWT_CYCCNTR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>DWT_CPICNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>DWT_SLPCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01C</td>
<td>DWT_PCSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>DWT_COMP0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>DWT_MASK0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>DWT_FUNCTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>DWT_COMP1R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x034</td>
<td>DWT_MASK1R</td>
</tr>
</tbody>
</table>

Reset value

<table>
<thead>
<tr>
<th>0x000</th>
<th>DWT_CTRLR</th>
<th>0x004</th>
<th>DWT_CYCCNTR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0x008</td>
<td>DWT_CPICNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x010</td>
<td>DWT_SLPCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01C</td>
<td>DWT_PCSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x020</td>
<td>DWT_COMP0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>DWT_MASK0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x028</td>
<td>DWT_FUNCTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>DWT_COMP1R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x034</td>
<td>DWT_MASK1R</td>
</tr>
</tbody>
</table>
Table 279. CPU2 DWT register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>0x038</th>
<th>0x040</th>
<th>0x044</th>
<th>0x048</th>
<th>0x050</th>
<th>0x054</th>
<th>0x058</th>
<th>0xFD0</th>
<th>0xFE0</th>
<th>0xFE4</th>
<th>0xFE8</th>
<th>0xFEC</th>
<th>0xFF0</th>
<th>0xFF4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DWT_FUNCT1R</td>
<td></td>
<td></td>
<td></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>DWT_COMP2R</td>
<td></td>
<td></td>
<td></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>DWT_MASK2R</td>
<td></td>
<td></td>
<td></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>DWT_FUNCT2R</td>
<td></td>
<td></td>
<td></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>DWT_COMP3R</td>
<td></td>
<td></td>
<td></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>DWT_MASK3R</td>
<td></td>
<td></td>
<td></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>DWT_FUNCT3R</td>
<td></td>
<td></td>
<td></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>DWT_PIDR4</td>
<td></td>
<td></td>
<td></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>DWT_PIDR0</td>
<td></td>
<td></td>
<td></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>DWT_PIDR1</td>
<td></td>
<td></td>
<td></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>DWT_PIDR2</td>
<td></td>
<td></td>
<td></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>DWT_PIDR3</td>
<td></td>
<td></td>
<td></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>DWT_CIDR0</td>
<td></td>
<td></td>
<td></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>DWT_CIDR1</td>
<td></td>
<td></td>
<td></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>

Reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x038</td>
<td>DWT_FUNCT1R</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x040</td>
<td>DWT_COMP2R</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x044</td>
<td>DWT_MASK2R</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x048</td>
<td>DWT_FUNCT2R</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x050</td>
<td>DWT_COMP3R</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>
</tr>
<tr>
<td>0x054</td>
<td>DWT_MASK3R</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x058</td>
<td>DWT_FUNCT3R</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0xFD0</td>
<td>DWT_PIDR4</td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>0xFE0</td>
<td>DWT_PIDR0</td>
<td>0 0 0 0 0 0 0 1 0</td>
</tr>
<tr>
<td>0xFE4</td>
<td>DWT_PIDR1</td>
<td>1 0 1 1 0 0 0 0</td>
</tr>
<tr>
<td>0xFE8</td>
<td>DWT_PIDR2</td>
<td>0 0 1 1 1 0 1 1</td>
</tr>
<tr>
<td>0xFEC</td>
<td>DWT_PIDR3</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0xFF0</td>
<td>DWT_CIDR0</td>
<td>0 0 0 0 1 1 0 0</td>
</tr>
<tr>
<td>0xFF4</td>
<td>DWT_CIDR1</td>
<td>0 1 1 1 0 0 0 0</td>
</tr>
</tbody>
</table>
Refer to Section 41.9: CPU2 ROM tables for the register boundary addresses.
41.11 CPU2 breakpoint unit (PBU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address and return a breakpoint instruction when a match is detected. The CPU2 PBU does not support Flash memory patch functionality.

41.11.1 BPU control register (BPU_CTRLR)

Address offset: 0x000
Reset value: 0x0000 0080

<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>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 14:12 NUM_CODE[6:4]: Number of instruction address comparators supported - least significant bits (read only).
0x0: 8 instruction comparators supported.

Bits 11:8 NUM_LIT[3:0]: Number of literal address comparators supported (read only).
0x0: No literal comparators supported.

Bits 7:4 NUM_CODE[3:0]: Number of instruction address comparators supported - least significant bits (read only).
0x8: 8 instruction comparators supported

Bit 1 KEY: Write protect key. A write to BPU_CTRLR register is ignored if this bit is not set to 1.

Bits 0 ENABLE: BPU enable
0x0: Disable
0x1: Enable

41.11.2 BPU remap register (BPU_REMAPR)

Address offset: 0x004
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 31:30 Reserved, must be kept at reset value.
Bit 29  **RMPSPT**: Indicates whether Flash memory patch remap is supported (read only).
0x0: Remapping not supported.

Bits 28:0  Reserved, must be kept at reset value.

### 41.11.3  **BPU comparator registers (BPU_COMPxR)**

Address offset: 0x008 + x * 0x4 (for x = 0 to 7)

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>INTORS</td>
<td>RES</td>
<td>COMP[26:14]</td>
<td></td>
<td></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></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>

**COM[13:0]**

<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>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</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>

**4KCOUNT[3:0]**  JEP106CON[3:0]

<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>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</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 31:30  **REPLACE[1:0]**: Defines the behavior when a match occurs between the COMP field and the instruction fetch address.
0x0: Reserved
0x1: Breakpoint on lower half-word, upper half-word is unaffected.
0x2: Breakpoint on upper half-word, lower half-word is unaffected.
0x3: Breakpoint on both upper and lower half-words.

Bit 29  Reserved, must be kept at reset value.

Bits 28:2  **COMP[26:0]**: Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF). If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1  Reserved, must be kept at reset value.

Bit 0  **ENABLE**: Comparator enable. The comparator is only enabled if both this bit and the BPU ENABLE bit in the BPU_CTRLR register are set.
0: Disabled
1: Enabled

### 41.11.4  **BPU CoreSight peripheral identity register 4 (BPU_PIDR4)**

Address offset: 0xFD0

Reset value: 0x0000 0004

<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>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</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 31:8  Reserved, must be kept at reset value.
41.11.5 **BPU CoreSight peripheral identity register 0 (BPU_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 000C

<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>
<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>
</tbody>
</table>

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

Bits 7:0: **PARTNUM[7:0]**: Part number bits [7:0]
0x0C: BPU part number

41.11.6 **BPU CoreSight peripheral identity register 1 (BPU_PIDR1)**

Address offset: 0xFE4
Reset value: 0x0000 00B0

<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>
<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>
</tbody>
</table>

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

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

0x0: BPU part number

41.11.7 **BPU CoreSight peripheral identity register 2 (BPU_PIDR2)**

Address offset: 0xFE8
Reset value: 0x0000 002B
Bit 31:8  Reserved, must be kept at reset value.

Bit 7:4  **REVISION[3:0]**: Component revision number
0x2: r0p3

Bit 3  **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

0x3: Arm® JEDEC code

### 41.11.8 BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

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

Bits 7:4  **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0  **CMOD[3:0]**: Customer modified
0x0: No customer modifications

### 41.11.9 BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFFFF
Reset value: 0x0000 0000D
41.11.10 **BPU CoreSight peripheral identity register 1 (BPU_CIDR1)**

Address offset: 0xFF4  
Reset value: 0x0000 00E0

<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>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>

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

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

41.11.11 **BPU CoreSight component identity register 2 (BPU_CIDR2)**

Address offset: 0xFF8  
Reset value: 0x0000 0005

<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>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>

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

Bits 7:4  **CLASS[3:0]**: Component ID bits [15:12] - component class  
0xE: Trace generator component  

0x0: Common ID value
Bits 7:0  **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

### 41.11.12  BPU CoreSight component identity register 3 (BPU_CIDR3)

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

<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, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value
## 41.11.13 CPU2 BPU register map and reset values

The CPU2 BPU registers are located at address range 0xE0002000 to 0xE0002FFC.

### Table 280. CPU2 BPU register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>0x000</th>
<th>0x004</th>
<th>0x008-0x024</th>
<th>0xFD0</th>
<th>0xFE0</th>
<th>0xFE4</th>
<th>0xFE8</th>
<th>0xFEC</th>
<th>0xFF0</th>
<th>0xFF4</th>
<th>0xFF8</th>
<th>0xFFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>BPU_CTRLR</td>
<td></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>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>BPU_REMAPR</td>
<td></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>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>0x008-0x024</td>
<td>BPU_COMP0-7R</td>
<td></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>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 41.9: CPU2 ROM tables for the register boundary addresses.
41.12 CPU2 cross trigger interface (CTI)

See Section 41.6.

41.13 CPU1 ROM table

The ROM table is a CoreSight™ component that contains the base addresses of all the Coresight™ debug components accessible via the AHB-AP. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one ROM table in the CPU1 sub-system. This table is pointed to by the BASE register in the CPU1 AHB-AP. It contains the base address pointer for the System control space registers, which allows the debugger to identify the CPU core, as well as for the FPB, DWT, ITM, ETM and CTI.

The CPU1 ROM table (see Table 281) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00FF000 to 0xE00FFFFC.

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00FF000</td>
<td>SCS</td>
<td>0xE000E000</td>
<td>0xFFF0F000</td>
<td>4 KB</td>
<td>0xFFF0F003</td>
</tr>
<tr>
<td>0xE00FF004</td>
<td>DWT</td>
<td>0xE0001000</td>
<td>0xFFF02000</td>
<td>4 KB</td>
<td>0xFFF02003</td>
</tr>
<tr>
<td>0xE00FF008</td>
<td>FPB</td>
<td>0xE0002000</td>
<td>0xFFF03000</td>
<td>4 KB</td>
<td>0xFFF03003</td>
</tr>
<tr>
<td>0xE00FF00C</td>
<td>ITM</td>
<td>0xE0000000</td>
<td>0xFFF01000</td>
<td>4 KB</td>
<td>0xFFF01003</td>
</tr>
<tr>
<td>0xE00FF010</td>
<td>TPIU</td>
<td>0xE0040000</td>
<td>0xFFF41000</td>
<td>4 KB</td>
<td>0xFFF41003</td>
</tr>
<tr>
<td>0xE00FF014</td>
<td>ETM</td>
<td>0xE0041000</td>
<td>0xFFF42000</td>
<td>4 KB</td>
<td>0xFFF42003</td>
</tr>
<tr>
<td>0xE00FF018</td>
<td>CTI</td>
<td>0xE0043000</td>
<td>0xFFF44000</td>
<td>4 KB</td>
<td>0xFFF44003</td>
</tr>
<tr>
<td>0xE00FF01C</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FF020 to 0xE00FFFC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FFCC to 0xE00FFFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 277</td>
</tr>
</tbody>
</table>

The topology for the CoreSight™ components in the CPU1 subsystem is shown in Figure 420.
Figure 420. CPU1 CoreSight™ topology

41.13.1 CPU1 ROM memory type register (C1ROM_MEMTYPEPER)

Address offset: 0x0000 0001
Reset value: 0x0000 0001
41.13.2 CPU1 ROM CoreSight peripheral identity register 4 (C1ROM_PIDR4)

Address offset: 0xFD0

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, must be kept at reset value.

Bit 0 **SYSMEM**: System memory

- 0x0: System memory is present on this bus

Bits 7:4 **4KCOUNT[3:0]**: register file size

- 0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

- 0x0: STMicroelectronics JEDEC continuation code

41.13.3 CPU1 ROM CoreSight peripheral identity register 0 (C1ROM_PIDR0)

Address offset: 0xFE0

Reset value: 0xFE000 0095

<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, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size

- 0x0: Register file occupies a single 4 KB region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

- 0x0: STMicroelectronics JEDEC continuation code

**PARTNUM[7:0]**
41.13.4  CPU1 ROM CoreSight peripheral identity register 1 (C1ROM_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 0004

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:0  | **PARTNUM[7:0]**: Part number bits [7:0] |
|          | 0x95: STM32WB55xx/35xx |

41.13.5  CPU1 ROM CoreSight peripheral identity register 2 (C1ROM_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 000A

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:4  | **JEP106ID[3:0]**: JEP106 identity code bits [3:0] |
|          | 0x0: STMicroelectronics JEDEC code |
|          | 0x4: STM32WB55xx/35xx |

|          | 0x2: STMicroelectronics JEDEC code |

1440/1530  RM0434 Rev 13
41.13.6 CPU1 ROM CoreSight peripheral identity register 3
(C1ROM_PIDR3)

Address offset: 0xFEC
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>

<table>
<thead>
<tr>
<th>REVAND[3:0]</th>
<th>CMOD[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:4 REVAND[3:0]: metal fix version
  0x0: No metal fix

Bits 3:0 CMOD[3:0]: Customer modified
  0x0: No customer modifications

41.13.7 CPU1 ROM CoreSight component identity register 0
(C1ROM_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

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

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

Bits 7:0 PREAMBLE[7:0]: Component ID bits [7:0]
  0x0D: Common ID value

41.13.8 CPU1 ROM CoreSight peripheral identity register 1
(C1ROM_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0010
41.13.9 CPU1 ROM CoreSight component identity register 2
(C1ROM_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

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

Bits 7:4 CLASS[3:0]: Component ID bits [15:12] - component class
0x1: ROM table component

0x0: Common ID value

41.13.10 CPU1 ROM CoreSight component identity register 3
(C1ROM_CIDR3)

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

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

Bits 7:0 PREAMBLE[19:12]: Component ID bits [23:16]
0x05: Common ID value
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value
### 41.13.11 CPU1 ROM table register map and reset values

#### Table 282. CPU1 ROM table 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  |
|--------|----------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xFCC  | C1ROM_MEMTYPEPER           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD0  | C1ROM_PIDR4                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE0  | C1ROM_PIDR0                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE4  | C1ROM_PIDR1                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE8  | C1ROM_PIDR2                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFEC  | C1ROM_PIDR3                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF0  | C1ROM_CIDR0                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | PREAMBLE[7:0]              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF4  | C1ROM_CIDR1                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | CLASS[3:0]                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | PREAMBLE[11:8]             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF8  | C1ROM_CIDR2                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | PREAMBLE[19:12]            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFFC  | C1ROM_CIDR3                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | PREAMBLE[27:20]            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 41.13: CPU1 ROM table for the register boundary addresses.
41.14 CPU1 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:
- Watchpoint
- ETM trigger, only available on STM32WB55xx
- PC sampling trigger
- Data address sampling trigger
- Data comparator (comparator 1 only)
- Clock cycle counter comparator (comparator 0 only)

It also contains counters for:
- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its DWT_COMPR register with one of the following:
- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:
- One or more DWT Data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value.
- A watchpoint debug event, on either the PC value or the accessed data address.
- A CMPMATCH[N] event, that signals the match outside the DWT unit.

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual [5].

41.14.1 DWT control register (DWT_CTRLR)

Address offset: 0x000
Reset value: 0x4000 0000
Bits 31:28 **NUMCOMP[3:0]**: Number of comparators implemented (read only)
   - 0x4: Four comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read only)
   - 0x0: Supported

Bit 26 **NOEXTRIG**: External match signal, CMPMATCH support (read only)
   - 0x0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read only)
   - 0x0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read only)
   - 0x0: Supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: Enable for POSTCNT underflow event counter packet generation
   - 0x0: Disabled
   - 0x1: Enabled

Bit 21 **FOLDEVTENA**: Enable for folded instruction counter overflow event generation
   - 0x0: Disabled
   - 0x1: Enabled

Bit 20 **LSUEVTENA**: Enable for LSU counter overflow event generation
   - 0x0: Disabled
   - 0x1: Enabled

Bit 19 **SLEEPETVENA**: Enable for sleep counter overflow event generation
   - 0x0: Disabled
   - 0x1: Enabled

Bit 18 **EXCEVTENA**: Enable for exception overhead counter overflow event generation
   - 0x0: Disabled
   - 0x1: Enabled

Bit 17 **CPIEVTENA**: Enable for CPI counter overflow event generation
   - 0x0: Disabled
   - 0x1: Enabled

Bit 16 **EXCTRCEVA**: Enable for exception trace generation
   - 0x0: Disabled
   - 0x1: Enabled

Bits 15:13 Reserved, must be kept at reset value.
Bit 12 PCSAMPLENA: Enables use of POSTCNT counter as a timer for Periodic PC sample packet generation.
  0x0: Disabled
  0x1: Enabled

Bits 11:10 SYNCTAP[1:0]: Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.
  0x0: Disabled. No synchronization packets
  0x1: Tap at CYCCNT[24]
  0x2: Tap at CYCCNT[26]
  0x3: Tap at CYCCNT[28]

Bit 9 CYCTAP: Selects the position of the POSTCNT tap on the CYCCNT counter.
  0x0: Tap at CYCCNT[6]
  0x1: Tap at CYCCNT[10]

Bits 8:5 POSTINIT[3:0]: Initial value of the POSTCNT counter. Writes to this field are ignored if POSTCNT counter is enabled (ie. CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 POSTPRESET[3:0]: Reload value of the POSTCNT counter.
  Bit 0 CYCCNTENA: Enables CYCCNT counter.
    0x0: Disabled
    0x1: Enabled

41.14.2 DWT cycle count register (DWT_CYCCNTR)
Address offset: 0x004
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>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</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 CYCCNT[31:0]: Processor clock cycle counter

41.14.3 DWT CPI count register (DWT_CPICNTR)
Address offset: 0x008
Reset value: 0x0000 0000
### DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **EXCCNT[7:0]**: Exception overhead cycle counter. Counts the number of cycles spent in exception processing.

### DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:8</th>
<th>Bit 7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **SLEEPCNT[7:0]**: Sleep cycle counter. Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).
41.14.6  **DWT LSU count register (DWT_LSUCNTR)**

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>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, must be kept at reset value.  
Bits 7:0  **LSUCNT[7:0]**: Load store counter. Counts additional cycles required to execute load and store instructions.

41.14.7  **DWT fold count register (DWT_FOLDCNTR)**

Address offset: 0x018  
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 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **FOLDCNT[7:0]**: Folded instruction counter. Increments on each instruction that takes 0 cycles.

41.14.8  **DWT program counter sample register (DWT_PCSR)**

Address offset: 0x01C  
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>EIASAMPLE[31:16]</td>
<td>EIASAMPLE[15:0]</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>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  **EIASAMPLE[31:0]**: Executed instruction address sample value. Samples the current value of the program counter.

### 41.14.9  DWT comparator register x (DWT_COMPxR)

Address offset: 0x020 + x * 0x10 (for x = 0 to 3)

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>

**COMP[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>

**COMP[15:0]**

<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>
</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>
</tr>
</tbody>
</table>

Bits 31:0  **COMP[31:0]**: Reference value for comparison.

### 41.14.10  DWT mask register x (DWT_MASKxR)

Address offset: 0x024 + x * 0x10 (for x = 0 to 3)

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>
</tr>
</thead>
</table>

**MASK[4:0]**

|     |     |     |     |     |     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  | rw  |

Bits 31:5  Reserved, must be kept at reset value.

Bits 4:0  **MASK[4:0]**: Comparator mask size. Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

### 41.14.11  DWT function register x (DWT_FUNCTxR)

Address offset: 0x028 + x * 0x10 (for x = 0 to 3)

Reset value: 0x0000 0000
### 41.14.12 DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

**Address offset**: 0xFD0

**Reset value**: 0x0000 0004
### 41.14.13 DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

**Address offset:** 0xFE0  
**Reset value:** 0x0000 0002

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>0x0000 0002</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>PARTNUM[7:0]</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>Bits 31:8 Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
| Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]  
0x02: DWT part number |

### 41.14.14 DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

**Address offset:** 0xFE4  
**Reset value:** 0x0000 00B9

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>0x0000 00B9</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>JEP106ID[3:0]</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>Bit 31:8 Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
41.14.15 **DWT CoreSight peripheral identity register 2 (DWT_PIDR2)**

Address offset: 0xFE8
Reset value: 0x0000 003B

<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>

Bit 31:8  Reserved, must be kept at reset value.
Bit s 7:4 **REVISION[3:0]**: Component revision number
  0x3: r0p4
Bit 3 **JEDEC**: JEDEC assigned value
  0x1: Designer ID specified by JEDEC
  0x3: Arm® JEDEC code

41.14.16 **DWT CoreSight peripheral identity register 3 (DWT_PIDR3)**

Address offset: 0xFEC
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>

Bit 31:8  Reserved, must be kept at reset value.
Bits 7:4 **REVAND[3:0]**: metal fix version
  0x0: No metal fix
Bits 3:0 **CMOD[3:0]**: Customer modified
  0x0: No customer modifications
### 41.14.17 DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<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, must be kept at reset value.  
Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]  
0x0D: Common ID value

### 41.14.18 DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 00E0

<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>

Bit 31:8 Reserved, must be kept at reset value.  
Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class  
0xE: Trace generator component  
0x0: Common ID value

### 41.14.19 DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8  
Reset value: 0x0000 0005

<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>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>
</tbody>
</table>
### 41.14.20 DWT CoreSight component identity register 3 (DWT_CIDR3)

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

<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, must be kept at reset value.

**Bits 7:0**  **PREAMBLE[19:12]**: Component ID bits [23:16]

- 0x05: Common ID value

<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, must be kept at reset value.

**Bits 7:0**  **PREAMBLE[27:20]**: Component ID bits [31:24]

- 0xB1: Common ID value
### 41.14.21 CPU1 DWT register map and reset values

The CPU1 DWT registers are located at address range 0xE0001000 to 0xE0001FFC.

<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>0x000</td>
<td>DWT_CTRLR</td>
<td>0x004</td>
<td>DWT_CYCCNTR</td>
<td>0x008</td>
<td>DWT_CPICNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
<td>0x010</td>
<td>DWT_SLPCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01C</td>
<td>DWT_PCSR</td>
<td>0x020</td>
<td>DWT_COMP0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>DWT_MASK0R</td>
<td>0x028</td>
<td>DWT_FUNCTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>DWT_COMP1R</td>
<td>0x034</td>
<td>DWT_MASK1R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x038</td>
<td>DWT_FUNCTOR1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 283. CPU1 DWT 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>0x000</td>
<td>DWT_CTRLR</td>
<td>0x004</td>
<td>DWT_CYCCNTR</td>
<td>0x008</td>
<td>DWT_CPICNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00C</td>
<td>DWT_EXCCNTR</td>
<td>0x010</td>
<td>DWT_SLPCCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x014</td>
<td>DWT_LSUCNTR</td>
<td>0x018</td>
<td>DWT_FOLDCNTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x01C</td>
<td>DWT_PCSR</td>
<td>0x020</td>
<td>DWT_COMP0R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x024</td>
<td>DWT_MASK0R</td>
<td>0x028</td>
<td>DWT_FUNCTOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x030</td>
<td>DWT_COMP1R</td>
<td>0x034</td>
<td>DWT_MASK1R</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x038</td>
<td>DWT_FUNCTOR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>0x040</td>
<td>DWT_COMP2R</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>
</tr>
<tr>
<td>0x044</td>
<td>DWT_MASK2R</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>
</tr>
<tr>
<td>0x048</td>
<td>DWT_FUNCT2R</td>
<td></td>
<td>MATCHED</td>
<td></td>
<td>DATAMATCH</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x050</td>
<td>DWT_COMP3R</td>
<td>COMP[31:0]</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>
</tr>
<tr>
<td>0x054</td>
<td>DWT_MASK3R</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>
</tr>
<tr>
<td>0x058</td>
<td>DWT_FUNCT3R</td>
<td></td>
<td>MATCHED</td>
<td></td>
<td>DATAMATCH</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFD0</td>
<td>DWT_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td>4KCOUNT[3:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFE0</td>
<td>DWT_PIDR0</td>
<td></td>
<td>PARTNUM[7:0]</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>
</tr>
<tr>
<td>0xFE4</td>
<td>DWT_PIDR1</td>
<td></td>
<td>JEP106ID[3:0]</td>
<td></td>
<td>PARTNUM[11:8]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>DWT_PIDR2</td>
<td></td>
<td>REVISION[3:0]</td>
<td></td>
<td>JEP106ID[6:4]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>DWT_PIDR3</td>
<td></td>
<td>REVAND[3:0]</td>
<td></td>
<td>CMODE[3:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFF0</td>
<td>DWT_CIDR0</td>
<td></td>
<td>PREAMBLE[7:0]</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>
</tr>
<tr>
<td>0xFF4</td>
<td>DWT_CIDR1</td>
<td></td>
<td>CLASS[3:0]</td>
<td></td>
<td>PREAMBLE[11:8]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0xFF8</td>
<td>DWT_CIDR2</td>
<td></td>
<td>PREAMBLE[19:12]</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>
</tr>
<tr>
<td>0xFFC</td>
<td>DWT_CIDR3</td>
<td></td>
<td>PREAMBLE[27:20]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Refer to Section 41.13: CPU1 ROM table for the register boundary addresses.
41.15 **CPU1 instrumentation trace macrocell (ITM)**

The ITM generates trace information as packets four sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. **Software trace**
   
   Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. **Hardware trace**
   
   The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparounds. The ITM outputs these packets on the trace bus.

3. **Local timestamping**
   
   The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

### 41.15.1 ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + x * 0x4 (x = 0 to 31)

Reset value: Unknown

<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>
</tbody>
</table>

Bits 31:0 **STIMULUS[31:0]**: Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FLFFOREADY indicator:

- 0: Stimulus port buffer is full (or port is disabled)
- 1: Stimulus port can accept new write data

### 41.15.2 ITM trace enable register (ITM_TER)

Address offset: 0x080

Reset value: 0x00000000
41.15.3 **ITM trace privilege register (ITM_TPR)**

Address offset: 0xE00

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>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 **PRIVMASK[31:0]**: Enable unprivileged access to ITM stimulus ports. Each bit controls eight stimulus ports.

- 0bXXX0: Unprivileged access permitted on ports 0 to 7
- 0bXXX1: Only privileged access permitted on ports 0 to 7
- 0bXX0X: Unprivileged access permitted on ports 8 to 15
- 0bXX1X: Only privileged access permitted on ports 8 to 15
- 0bX0XX: Unprivileged access permitted on ports 16 to 23
- 0bX1XX: Only privileged access permitted on ports 16 to 23
- 0b0XXX: Unprivileged access permitted on ports 24 to 31
- 0b1XXX: Only privileged access permitted on ports 24 to 31

41.15.4 **ITM trace control register (ITM_TCR)**

Address offset: 0xE80

Reset value: 0x0000 0000
### 41.15.5 ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0  
Reset value: 0x0000 0004
### Debug support (DBG)

#### ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0  
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>

<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, must be kept at reset value.

Bits 7:4 **4KCOUNT[3:0]**: register file size  
0x0: Register file occupies a single 4 KB region

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

#### ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00B0

<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>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, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]  
0x01: ITM part number

Bit 31:8 Reserved, must be kept at reset value.
41.15.8 ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 003B

<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>
</tbody>
</table>

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

Bits 7:4 REVISON[3:0]: Component revision number
0x3: r0p4

Bit 3 JEDEC: JEDEC assigned value
0x1: Designer ID specified by JEDEC

0x3: Arm® JEDEC code

41.15.9 ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC
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>
</tbody>
</table>

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

Bits 7:4 REVAND[3:0]: metal fix version
0x0: No metal fix

Bits 3:0 CMOD[3:0]: Customer modified
0x0: No customer modifications
41.15.10  ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<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>

PREAMBLE[7:0]

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  PREAMBLE[7:0]: Component ID bits [7:0]
0x0D: Common ID value

41.15.11  ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 00E0

<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>


Bit 31:8  Reserved, must be kept at reset value.
Bits 7:4  CLASS[3:0]: Component ID bits [15:12] - component class
0xE: Trace generator component
0x0: Common ID value

41.15.12  ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005
### 41.15.13 ITM CoreSight component identity register 3 (ITM_CIDR3)

**Address offset:** 0xFFC

**Reset value:** 0x0000 00B1

<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, must be kept at reset value.

**Bits 7:0**  **PREAMBLE[19:12]**: Component ID bits [23:16]

0x05: Common ID value

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

**Bits 7:0**  **PREAMBLE[27:20]**: Component ID bits [31:24]

0xB1: Common ID value
### 41.15.14 ITM register map and reset values

The ITM registers are located at address range 0xE0000000 to 0xE000FFC.

Table 284. CPU1 ITM 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   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | ITM_STIM0-31R |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x000  | ITM_TER       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x00E0 | ITM_TPR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x00E0 | ITM_TCR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0FE0 | ITM_PIDR0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0FE4 | ITM_PIDR1     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0FE8 | ITM_PIDR2     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFF0 | ITM_CIDR0     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFF4 | ITM_CIDR1     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFF8 | ITM_CIDR2     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0xFFF0| ITM_CIDR3     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Refer to Section 41.13: CPU1 ROM table for the register boundary addresses.
41.16 **CPU1 breakpoint unit (FPB)**

The FPB allows the user to set hardware breakpoints. It contains six comparators that monitor the instruction fetch address and two literal address comparators. If a match occurs, the address is remapped to an address in system memory, defined by the FPB_REMAPR register plus an offset corresponding to the matching comparator. Alternatively, the instruction comparators can be configured to generate a breakpoint instruction.

### 41.16.1 FPB control register (FPB_CTRLR)

Address offset: 0x000

Reset value: 0x0000 0260

<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>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 14:12 **NUM_CODE[6:4]**: Number of instruction address comparators supported - least significant bits (read only).

- 0x0: 6 instruction comparators supported.

Bits 11:8 **NUM_LIT[3:0]**: Number of literal address comparators supported (read only).

- 0x2: Two literal comparators supported.

Bits 7:4 **NUM_CODE[3:0]**: Number of instruction address comparators supported - least significant bits (read only).

- 0x6: 6 instruction comparators supported

Bit 1 **KEY**: Write protect key. A write to FPB_CTRLR register is ignored if this bit is not set to 1.

Bits 0 **ENABLE**: FPB enable

- 0x0: Disable
- 0x1: Enable

### 41.16.2 FPB remap register (FPB_REMAPR)

Address offset: 0x004

Reset value: 0x2000 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>r</td>
<td>rw</td>
<td>rw</td>
<td>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>
<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 **RMPSPX**: Indicates whether Flash memory patch remap is supported (read only).

- 0x1: Remapping supported.

Bits 28:5 **REMAP[23:0]**: Remap target address. Bits [28:5] of the base address in SRAM to which the FPB remaps the address. The remap base address must be aligned to the number of words required to support the implemented comparators, that is to (NUM_CODE+NUM_LIT) words, with a minimum alignment of 8 words. Because remap is into the SRAM memory region, 0x20000000-0x3FFFFFFF, bits [31:29] of the remap address are 0b001.

Bits 4:0  Reserved, must be kept at reset value.

### 41.16.3 FPB comparator registers (FPB_COMPxR)

Address offset: 0x0008 + x * 0x4 (for x = 0 to 7)

Reset value: 0x0000 0000

| \(31\) | \(30\) | \(29\) | \(28\) | \(27\) | \(26\) | \(25\) | \(24\) | \(23\) | \(22\) | \(21\) | \(20\) | \(19\) | \(18\) | \(17\) | \(16\) |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COMP[13:0] | Res. | ENABLE |
| \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) | \(\text{rw}\) |

Bits 31:30  **REPLACE[1:0]**: Defines the behavior when a match occurs between the COMP field and the instruction fetch address.

- 0x0: Reserved
- 0x1: Breakpoint on lower half-word, upper half-word is unaffected.
- 0x2: Breakpoint on upper half-word, lower half-word is unaffected.
- 0x3: Breakpoint on both upper and lower half-words.

Bit 29  Reserved, must be kept at reset value.

Bits 28:2  **COMP[26:0]**: Value to compare with address bits 28:2 of accesses to instruction code memory (0x00000000 to 0x1FFFFFFF). If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1  Reserved, must be kept at reset value.

Bit 0  **ENABLE**: Comparator enable. The comparator is only enabled if both this bit and the FPB ENABLE bit in the FPB_CTRLR register are set.

- 0: Disabled
- 1: Enabled

### 41.16.4 FPB CoreSight peripheral identity register 4 (FPB_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004
### 41.16.5 FPB CoreSight peripheral identity register 0 (FPB_PIDR0)

**Address offset:** 0xFE0  
**Reset value:** 0x0000 0003

<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, must be kept at reset value.  

Bits 7:4 **4KCOUNT[3:0]:** register file size  
- 0x0: Register file occupies a single 4 KB region  

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

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

### 41.16.6 FPB CoreSight peripheral identity register 1 (FPB_PIDR1)

**Address offset:** 0xFE4  
**Reset value:** 0x0000 00B0

<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, must be kept at reset value.  

Bits 7:0 **PARTNUM[7:0]:** Part number bits [7:0]  
- 0x03: FPB part number

*Bit 31:8 Reserved, must be kept at reset value.*
Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
- 0xB: Arm® JEDEC code

- 0x0: FPB part number

### 41.16.7 FPB CoreSight peripheral identity register 2 (FPB_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 002B

<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>09</td>
<td>08</td>
<td>07</td>
<td>06</td>
<td>05</td>
<td>04</td>
<td>03</td>
<td>02</td>
<td>01</td>
<td>00</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>

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

Bit s 7:4 **REVISION[3:0]**: Component revision number
- 0x2: r0p3

Bit 3 **JEDEC**: JEDEC assigned value
- 0x1: Designer ID specified by JEDEC

- 0x3: Arm® JEDEC code

### 41.16.8 FPB CoreSight peripheral identity register 3 (FPB_PIDR3)

Address offset: 0xFEC
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>09</td>
<td>08</td>
<td>07</td>
<td>06</td>
<td>05</td>
<td>04</td>
<td>03</td>
<td>02</td>
<td>01</td>
<td>00</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>

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

Bits 7:4 **REVAND[3:0]**: metal fix version
- 0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
- 0x0: No customer modifications
### 41.16.9 FPB CoreSight component identity register 0 (FPB_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| **PREAMBLE[7:0]** |

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]

0x0D: Common ID value

### 41.16.10 FPB CoreSight peripheral identity register 1 (FPB_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 00E0

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| **CLASS[3:0]** | **PREAMBLE[11:8]** |

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

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class

0xE: Trace generator component


0x0: Common ID value

### 41.16.11 FPB CoreSight component identity register 2 (FPB_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005
### 41.16.12 FPB CoreSight component identity register 3 (FPB_CIDR3)

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

<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>PREAMBLE[19:12]</td>
<td></td>
<td></td>
<td></td>
<td></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>
<tr>
<td>PREAMBLE[27:20]</td>
<td></td>
<td></td>
<td></td>
<td></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  **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

**0xB1: Common ID value**
41.16.13 FPB register map and reset values

The CPU1 FPB registers are located at address range 0xE0002000 to 0xE0002FFC.

Table 285. CPU1 FPB 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>0x000</td>
<td>FPB_CTRLR</td>
<td>0x004</td>
<td>FPB_REMAPR</td>
<td>0x008</td>
<td>FPB_COMP0-7R</td>
</tr>
<tr>
<td>0x004</td>
<td>FPB_REMAPR</td>
<td>0x008</td>
<td>FPB_COMP0-7R</td>
<td>0xF04</td>
<td>FPB_CIDR0</td>
</tr>
<tr>
<td>0xF04</td>
<td>FPB_CIDR0</td>
<td>0xFF0</td>
<td>FPB_CIDR3</td>
<td>0xFF4</td>
<td>FPB_CIDR3</td>
</tr>
<tr>
<td>0xFF4</td>
<td>FPB_CIDR3</td>
<td>0xFFF</td>
<td>FPB_CIDR2</td>
<td>0xFFF</td>
<td>FPB_CIDR2</td>
</tr>
</tbody>
</table>

Refer to Section 41.13: CPU1 ROM table for the register boundary addresses.
41.17 **CPU1 Embedded trace macrocell (ETM™), only available on STM32WB55xx**

The CPU1 ETM™ is a CoreSight™ component closely coupled to the CPU. The ETM™ generates trace packets that enable to trace the execution of the CPU1 core. It is configured for instruction trace only, i.e. data accesses are not included in the trace information.

The ETM™ receives information from the CPU over the processor trace interface, including:
- The number of instructions executed in the same cycle
- Changes in program flow
- The current processor instruction state
- The addresses of memory locations accessed by load and store instructions
- The type, direction and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM™-Cortex®-M4 technical reference manual.

### 41.17.1 ETM control register (ETM_CR)

Address offset: 0x000
Reset value: 0x0000 0411

<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></td>
<td></td>
<td></td>
<td></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></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>
</tbody>
</table>

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

Bit 27  **TSEN**: Timestamp Enable.
- 0: Timestamping disabled
- 1: Timestamping enabled

Bits 26:12  Reserved, must be kept at reset value.

Bit 11  **ETMEN**: ETM enable.
- 0: Trace output disabled
- 1: Trace output enabled

Bit 10  **PROG**: ETM programming. This bit must be set to 1 before programming the ETM™. Tracing is prevented while this bit is set to 1.
- 0x0: ETM operational
- 0x1: ETM in programming state
Bit 9 **BO**: Branch output. When set to 1 all branch addresses are output, even if the branch was because of a direct branch instruction. Setting this bit enables reconstruction of the program flow without having access to the memory image of the code being executed.

When this bit is set to 1, more trace data is generated, and this may affect the performance of the trace system. Information about the execution of a branch is traced regardless of the state of this bit.

Bit 8 **STALL**: Stall processor. The FIFOFULL output can be used to stall the processor to prevent overflow. The FIFOFULL output is only enabled when the stall processor bit is set to 1. When the bit is 0 the FIFOFULL output remains LOW at all times and the FIFO overflows if there are too many trace packets. Trace resumes without corruption once the FIFO has drained, if overflow does occur.

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **PDN**: ETM power down. This bit can be used by an implementation to control if the ETM is in a low power state. This bit must be cleared by the trace software tools at the beginning of a debug session.

When this bit is set to 1, writes to some registers and fields might be ignored. You can always write to the following registers and fields:
- ETMCR bit [0]
- ETMLAR
- ETMCLAIMSET register
- ETMCLAIMCLR register

When the ETMCR is written with this bit set to 1, bits other than bit [0] might be ignored.

### 41.17.2 ETM configuration code register (ETM_CCR)

**Address offset**: 0x004

**Reset value**: 0x8C80 2000

<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></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></td>
</tr>
</tbody>
</table>

**Bit 30** **IDPRES**: ETM ID register present.
1: ETMIDR register is present and defines the ETM architecture version in use.

**Bits 30:28** Reserved, must be kept at reset value.

**Bit 27** **CPMAC**: Co-processor and memory access.
1: Memory-mapped access to registers is supported.

**Bit 26** **TSPRES**: Trace start/stop block present.
1: Trace start/stop block is present.

**Bits 25:24** **CIDCMP[1:0]**: Number of context ID comparators.
0x0: Context ID comparators are not implemented.
Bit 23 **FFPRES**: FIFOFULL logic present. To use FIFOFULL the system must also support the function, as indicated by bit[8] of ETMSCR.
1: FIFOFULL logic is present in the ETM.

Bits 22:20 **NEXTO[2:0]**: Number of external outputs.
0x0: No external outputs are supported

Bits 19:17 **NEXTI[2:0]**: Number of external inputs.
0x2: Two external inputs are supported

Bit 16 **SEQPRES**: Sequencer present.
0: The sequencer is not present in the ETM.

Bits 15:13 **NCNT[2:0]**: Number of counters.
0x1: One counter is implemented

Bits 12:8 **NMMDEC[4:0]**: Number of memory map decoders.
0x0: No memory map decoder inputs are implemented.

Bits 7:4 **NDVCMP[3:0]**: Number of data value comparators.
0x0: No data value comparators are implemented.

Bits 3:0 **NADCMP[3:0]**: Number of address comparator pairs.
0x0: No address comparator pairs are implemented.

### 41.17.3 ETM trigger register (ETM_TRIGGER)

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>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</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>

**FCN[1:0]**

**RESOURCESB[6:0]**

**RESOURCESA[6:0]**

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

Bits 16:14 **FCN[2:0]**: Boolean function. If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.
0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier. Bits [6:4] define the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

**Type (bits [6:4]) Index (bits [3:0])**

- 0x2 0-3: Embedded-ICE watch point comparators 1-4 (from DWT)
- 0x4 0: Counter 1 at zero
- 0x5 15: Trace start/stop resource
- 0x6 0-1: External inputs 1-2
- 0x6 15: Hard-wired input, always true

### 41.17.4 ETM status register (ETM_SR)

Address offset: 0x010

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>rw</td>
<td>rw</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 31:4 Reserved, must be kept at reset value.

- **Bit 3 TRIGFL**: Trigger flag. Set when the trigger occurs, and prevents the trigger from being output until the ETM is programmed again.

- **Bit 2 TSSRSTAT**: Trigger start/stop resource status. Holds the current status of the trace start/stop resource. If set to 1, it indicates that a trace on address has been matched, without a corresponding trace off address match.

- **Bit 1 PROGVAL**: Programming bit value (read only). The current effective value of the ETM Programming bit, bit [10] of the ETM_CR. You must set the programming bit to 1 and wait for this bit to go to 1 before you start to program the ETM.

  This bit remains at 0 until the FIFO is empty, or if OS lock is set in the ETM_OSLSR register.

- **Bit 0 UOVFL**: Untraced overflow (read only). If set to 1, there is an overflow that has not yet been traced. This bit is cleared to 0 when trace is restarted, or when the ETM power down bit (bit [0] of the ETM_CR register) is set to 1.

### 41.17.5 ETM status register (ETM_SCR)

Address offset: 0x014

Reset value: 0x0002 0D09
### Debug support (DBG)

#### 41.17.6 ETM trace enable event register (ETM_TEEVR)

Address offset: 0x020

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17</td>
<td>NOFTCHCOMP: No fetch comparisons.</td>
</tr>
<tr>
<td></td>
<td>1: Fetch comparisons are not implemented</td>
</tr>
<tr>
<td>Bit 16-15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 14-12</td>
<td>NSUPPROC[2:0]: Number of supported processors.</td>
</tr>
<tr>
<td></td>
<td>0x0: One processor supported.</td>
</tr>
<tr>
<td>Bit 11</td>
<td>PORTMODESUP: Port mode support.</td>
</tr>
<tr>
<td></td>
<td>0: Current selected port mode is not supported.</td>
</tr>
<tr>
<td></td>
<td>1: Current selected port mode is supported.</td>
</tr>
<tr>
<td>Bit 10</td>
<td>PORTSIZESUP: Port size support.</td>
</tr>
<tr>
<td></td>
<td>0: Current selected port size is not supported.</td>
</tr>
<tr>
<td></td>
<td>1: Current selected port size is supported.</td>
</tr>
<tr>
<td>Bit 9</td>
<td>MAXPORTSIZE[3]: Maximum ETM port size bit [3]. This bit is used in conjunction with bits [2:0]</td>
</tr>
<tr>
<td>Bit 8</td>
<td>FFSUP: FIFOFULL support.</td>
</tr>
<tr>
<td></td>
<td>1: FIFOFULL is supported.</td>
</tr>
<tr>
<td>Bit 7-3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 2-0</td>
<td>MAXPORTSIZE[2:0]: Maximum ETM port size bit [2:0]. These bits in conjunction with bit 9 indicate the maximum ETM port size supported.</td>
</tr>
<tr>
<td></td>
<td>0x1: Maximum port size = 1</td>
</tr>
</tbody>
</table>

#### ETM trace enable event register (ETM_TEEVR)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSUPPROC[2:0]</td>
<td>Number of supported processors.</td>
<td>0x0: One processor supported.</td>
</tr>
<tr>
<td>PORTMODESUP</td>
<td>Port mode support.</td>
<td>0: Current selected port mode is not supported.</td>
</tr>
<tr>
<td>PORTSIZESUP</td>
<td>Port size support.</td>
<td>0: Current selected port size is not supported.</td>
</tr>
<tr>
<td>MAXPORTSIZE[3]</td>
<td>Maximum ETM port size bit [3]. This bit is used in conjunction with bits [2:0]</td>
<td></td>
</tr>
<tr>
<td>FFSUP</td>
<td>FIFOFULL support.</td>
<td>1: FIFOFULL is supported.</td>
</tr>
<tr>
<td>MAXPORTSIZE[2:0]</td>
<td>Maximum ETM port size bit [2:0]. These bits in conjunction with bit 9 indicate the maximum ETM port size supported.</td>
<td></td>
</tr>
<tr>
<td>NOFTCHCOMP</td>
<td>No fetch comparisons.</td>
<td>1: Fetch comparisons are not implemented</td>
</tr>
</tbody>
</table>

Address offset: 0x020

Reset value: 0x0000 0000
41.17.7 ETM trace enable control 1 register (ETM_TECR1)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **ENONOFF**: Trace start/stop trace enable control.
0: Trace start/stop block does not control trace enable.
1: Trace enable is controlled by the trace start/stop block.

Bits 24:0 Reserved, must be kept at reset value.

41.17.8 ETM FIFOFULL level register (ETM_FFLR)

Address offset: 0x028

Reset value: 0x0000 0000
## 41.17.9  ETM counter reload value 1 register (ETM_CNTRLDVR1)

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></td>
<td></td>
<td></td>
<td></td>
<td></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>

**LEVEL[7:0]**  
Bits 31:8  Reserved, must be kept at reset value.  
Bit7:0 **LEVEL[7:0]**: The number of bytes left in the FIFO, below which the FIFOFULL signal is asserted. For example, setting this value to 15 causes processor stalling, if enabled, when there are less than 15 free bytes in the FIFO.

<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>

**INICNT[15:0]**  
Bits 31:16  Reserved, must be kept at reset value.  
Bit15:0 **INICNT[15:0]**: Initial count. Specifies the counter reload value.

## 41.17.10  ETM synchronization frequency register (ETM_SYNCFR)

Address offset: 0x1E0  
Reset value: 0x0000 0400

<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>

**FREQUENCY[11:0]**  
Bits 31:12  Reserved, must be kept at reset value.  
Bits 11:0 **FREQUENCY[11:0]**: This value is the time in bytes between synchronization points in the trace (the tools can start decompressing only at synchronization points).  
0x400: The ETM uses a fixed synchronization packet generation frequency of every 1024 bytes of trace.
### 41.17.11 ETM ID register (ETM_IDR)

Address offset: 0x1E4  
Reset value: 0x4114 F250

<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>f</td>
<td>r</td>
<td>r</td>
<td>f</td>
<td>f</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Bits 31:24** DESIGNER[7:0]: Implementer code.  
  0x41: ARM®
- **Bits 23:21** Reserved, must be kept at reset value.
- **Bit 20** ABPE: Alternative ranch packet encoding.  
  1: ABPE is implemented.
- **Bit 19** SXSUPP: Security extensions support.  
  0: ETM behaves as if the processor is in secure mode at all times.
- **Bit 18** T32SUPP: 32-bit Thumb instruction tracing.  
  1: 32-bit Thumb instructions are traced as single instructions.
- **Bit 17** Reserved, must be kept at reset value.
- **Bit 16** LDPCF: Load PC first.  
  0: Data tracing is not supported
- **Bits 15:12** PROCFAI[3:0]: Processor family.  
  0xF: Processor family is not identified in this register
- **Bits 11:8** ETMARCHMAJ[3:0]: Major ETM architecture version.  
  0x2: Version 3
- **Bits 7:4** ETMARCHMIN[3:0]: Minor ETM architecture version.  
  0x5: Version 5
- **Bits 3:0** REVISION[3:0]: Implementation revision.  
  0x0: Revision 0

### 41.17.12 ETM configuration code extension register (ETM_CCER)

Address offset: 0x1E8  
Reset value: 0x1854 1800
### Debug support (DBG)

**ETM trace enable start/stop EmbeddedICE control register (ETM_TESSEICR)**

Address offset: 0x1F0

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:30</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 29</td>
<td><strong>TSSIZE</strong>: Timestamp size.</td>
</tr>
<tr>
<td></td>
<td>0: 48 bits.</td>
</tr>
<tr>
<td>Bit 28</td>
<td><strong>TSENC</strong>: Timestamp encoding.</td>
</tr>
<tr>
<td></td>
<td>1: Timestamp is encoded as a natural binary number.</td>
</tr>
<tr>
<td>Bit 27</td>
<td><strong>RFC</strong>: Reduced function counter.</td>
</tr>
<tr>
<td></td>
<td>1: Counter is a reduced function counter.</td>
</tr>
<tr>
<td>Bits 26:23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 22</td>
<td><strong>TSIMP</strong>: Timestamping implemented.</td>
</tr>
<tr>
<td></td>
<td>1: Timestamping implemented</td>
</tr>
<tr>
<td>Bit 21</td>
<td><strong>EIIMP</strong>: EmbeddedICE behavior control implemented.</td>
</tr>
<tr>
<td></td>
<td>0: Not implemented</td>
</tr>
<tr>
<td>Bit 20</td>
<td><strong>TSSUWP</strong>: Trace start/stop uses Embedded ICE watchpoint inputs.</td>
</tr>
<tr>
<td></td>
<td>1: Yes</td>
</tr>
<tr>
<td>Bits 19:16</td>
<td><strong>NUMWPIN[3:0]</strong>: Embedded ICE watchpoint inputs. Number of inputs coming from the DWT.</td>
</tr>
<tr>
<td></td>
<td>0x4: Four inputs</td>
</tr>
<tr>
<td>Bits 15:13</td>
<td><strong>NUMIR[2:0]</strong>: Instrumentation resources.</td>
</tr>
<tr>
<td></td>
<td>0x0: None</td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>SUPPDAC</strong>: Data address comparison support.</td>
</tr>
<tr>
<td></td>
<td>1: Not supported</td>
</tr>
<tr>
<td>Bit 11</td>
<td><strong>RR</strong>: Readable registers.</td>
</tr>
<tr>
<td></td>
<td>1: All registers are readable</td>
</tr>
<tr>
<td>Bits 10:3</td>
<td><strong>XXINBUS[7:0]</strong>: Extended external input bus.</td>
</tr>
<tr>
<td></td>
<td>0x0: Not implemented</td>
</tr>
<tr>
<td>Bits 2:0</td>
<td><strong>XXINSEL[2:0]</strong>: Extended external input selectors.</td>
</tr>
<tr>
<td></td>
<td>0x0: Not implemented</td>
</tr>
</tbody>
</table>

---

**Address offset:** 0x1530

**Reset value:** 0x0000 0000

---
### 41.17.14 ETM timestamp event register (ETM_TSEVR)

Address offset: 0x1F8  
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: Reserved, must be kept at reset value.


<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>FCN [1:0]</td>
<td>RESOURCEB[6:0]</td>
<td>RESOURCEA[6:0]</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>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<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 31:17: Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]:** Boolean function. If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

- 0x0: A  
- 0x1: NOT A  
- 0x2: A AND B  
- 0x3: NOT A AND B  
- 0x4: NOT A AND NOT B  
- 0x5: A OR B  
- 0x6: NOT A OR B  
- 0x7: NOT A OR NOT B


Bits 6:0 **RESOURCEA[6:0]:** First resource identifier. Bits [6:4] define the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
41.17.15 **ETM trace ID register (ETM_TRACEIDR)**

Address offset: 0x200
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Type (bits [6:4])</th>
<th>Index (bits [3:0])</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2</td>
<td>0-3</td>
</tr>
<tr>
<td>0x4</td>
<td>0</td>
</tr>
<tr>
<td>0x5</td>
<td>15</td>
</tr>
<tr>
<td>0x6</td>
<td>0-1</td>
</tr>
</tbody>
</table>

- Embedded-ICE watchpoint comparators 1-4 (from DWT)
- Counter 1 at zero
- Trace start/stop resource
- External inputs 1-2
- Hard-wired input, always true

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace ID to output onto the trace bus. This should be programmed with a unique value to differentiate it from other trace sources in the system.

41.17.16 **ETM ID register 2(ETM_IDR2)**

Address offset: 0x208
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Type (bits [6:2])</th>
<th>Index (bits [1:0])</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2</td>
<td>0-3</td>
</tr>
<tr>
<td>0x4</td>
<td>0</td>
</tr>
<tr>
<td>0x5</td>
<td>15</td>
</tr>
<tr>
<td>0x6</td>
<td>0-1</td>
</tr>
</tbody>
</table>

- Embedded-ICE watchpoint comparators 1-4 (from DWT)
- Counter 1 at zero
- Trace start/stop resource
- External inputs 1-2
- Hard-wired input, always true

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

Bit 1 **SWPTO**: Identifies the order for a SWP or SWPB instruction.
- 0: The Load transfer is traced before the Store transfer

Bit 0 **RFETO**: Identifies the order for a RFE instruction.
- 0: The PC transfer is traced before the CPSR transfer

41.17.17 **ETM device power down status register 2(ETM_PDSR)**

Address offset: 0x314
### 41.17.18 ETM claim tag set register (ETM_CLAIMSETR)

Address offset: 0xFA0  
Reset value: 0x0000 000F

<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 CLAIMSET[3:0]: Set claim tag bits  
Write:  
- 0000: No effect  
- xxx1: Set bit 0  
- xx1x: Set bit 1  
- x1xx: Set bit 2  
- 1xxx: Set bit 3  
Read:  
- 0xF: Indicates there are four bits in claim tag

### 41.17.19 ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4  
Reset value: 0x0000 0000
Debug support (DBG)

41.17.20 ETM lock access register (ETM_LAR)

Address offset: 0xFB0
Reset value: N/A

<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 ACCESS_W[31:0]: Enables write access to some ETM registers by processor cores (debuggers do not need to unlock the component)
- 0xC5ACCE55: Enable write access
- Other values: Disable write access

41.17.21 ETM lock status register (ETM_LSR)

Address offset: 0xFB4
Reset value: 0x0000 0003

<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 31:15 LOCK_TYPE: Lock type
- 0: None
- 1: Grant
- 2: Unlock
- 3: Lock

The document provides detailed information about the registers, their addresses, reset values, and bit descriptions.
Bits 31:3 Reserved, must be kept at reset value.

Bit 2  **LOCKTYPE**: Indicates the size of the ETM_LAR register
   0: 32-bit

Bit 1  **LOCKGRANT**: Current status of lock. This bit always reads as zero by an external debugger.
   0: Write access is permitted
   1: Write access is blocked. Only reads are permitted.

Bit 0  **LOCKEXIST**: Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.
   0: No lock control mechanism exists.
   1: Lock control mechanism is implemented

### 41.17.22 ETM authentication status register (ETM_AUTHSTATR)

Address offset: 0xFB8  
Reset value: 0x0000 000A

<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>

<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>SNID[1:0]</td>
<td>SID[1:0]</td>
<td>NSNID[1:0]</td>
<td>NSID[1:0]</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>
</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 31:8 Reserved, must be kept at reset value.

Bits 7:6  **SNID[1:0]**: Security level for secure non-invasive debug  
0x0: Not implemented

Bits 5:4  **SID[1:0]**: Security level for secure invasive debug  
0x0: Not implemented

Bits 3:2  **NSNID[1:0]**: Security level for non-secure non-invasive debug  
0x0: Not implemented

Bits 1:0  **NSID[1:0]**: Security level for non-secure invasive debug  
0x0: Not implemented

### 41.17.23 ETM CoreSight device identity register (ETM_DEVTYPEP)

Address offset: 0xFCC  
Reset value: 0x0000 0013

<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>

<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>SUBTYPE[3:0]</td>
<td>MAJORTYPE[3:0]</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>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>

1486/1530  
RM0434 Rev 13
41.17.24 **ETM CoreSight peripheral identity register 4 (ETM_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0004

<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>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 31:8  Reserved, must be kept at reset value.  
Bits 7:4  **4KCOUNT[3:0]**: register file size  
0x0: Register file occupies a single 4 KB region  

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

41.17.25 **ETM CoreSight peripheral identity register 0 (ETM_PIDR0)**

Address offset: 0xFE0  
Reset value: 0x0000 0025

<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>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 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **PARTNUM[7:0]**: Part number bits [7:0]  
0x25: ETM part number

41.17.26 **ETM CoreSight peripheral identity register 1 (ETM_PIDR1)**

Address offset: 0xFE4  
Reset value: 0x0000 00B9
41.17.27 ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 002B

Bits 7:4 \textbf{JEP106ID[3:0]}: JEP106 identity code bits [3:0]
0xB: 	extsuperscript{®} JEDEC code

0x9: ETM part number

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

41.17.28 ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

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

Bit s 7:4 \textbf{REVISION[3:0]}: Component revision number
0x0: r0p1

Bit 3 \textbf{JEDEC}: JEDEC assigned value
0x1: Designer ID specified by JEDEC

0x3: Arm	extsuperscript{®} JEDEC code
Bit 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
- 0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
- 0x0: No customer modifications

**41.17.29 ETM CoreSight component identity register 0 (ETM_CIDR0)**

Address offset: 0xFF0
Reset value: 0x0000 000D

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
- 0x0D: Common ID value

**41.17.30 ETM CoreSight peripheral identity register 1 (ETM_CIDR1)**

Address offset: 0xFF4
Reset value: 0x0000 0090

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

Bits 7:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
Bit 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]**: Component ID bits [15:12] - component class
         0x9: Trace generator component

         0x0: Common ID value

### 41.17.31 ETM CoreSight component identity register 2 (ETM_CIDR2)

*Address offset: 0xFF8*

*Reset value: 0x0000 0005*

<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, must be kept at reset value.

Bits 7:0  **PREAMBLE[19:12]**: Component ID bits [23:16]
         0x05: Common ID value

### 41.17.32 ETM CoreSight component identity register 3 (ETM_CIDR3)

*Address offset: 0xFFC*

*Reset value: 0x0000 00B1*

<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, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: Component ID bits [31:24]
         0xB1: Common ID value
## 41.17.33 ETM register map and reset values

The ETM registers are accessed by the debugger via the CPU1 AHB-AP, at address range 0x0E0041000 to 0x0E0041FFC.

### Table 286. CPU1 ETM register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>ID</th>
<th>Type</th>
<th>Range</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>ETM_CR</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x004</td>
<td>ETM_CCR</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0x008</td>
<td>ETM_TRIGGER</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x010</td>
<td>ETM_SR</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x014</td>
<td>ETM_SCR</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0x020</td>
<td>ETM_TEEVR</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x024</td>
<td>ETM_TECR1</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x028</td>
<td>ETM_FFLR</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x140</td>
<td>ETM_CNTRLDVR1</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x1E0</td>
<td>ETM_SYNCFR</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x1E4</td>
<td>ETM_IDR</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Reset value**

0x000  ETM_CR
0x004  ETM_CCR
0x008  ETM_TRIGGER
0x010  ETM_SR
0x014  ETM_SCR
0x020  ETM_TEEVR
0x024  ETM_TECR1
0x028  ETM_FFLR
0x140  ETM_CNTRLDVR1
0x1E0  ETM_SYNCFR
0x1E4  ETM_IDR
#### Table 286. CPU1 ETM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>TSSIZE</th>
<th>TSENC</th>
<th>RRF</th>
<th>SSTMP</th>
<th>TSSJVP</th>
<th>NUMWPN[3:0]</th>
<th>NUMMI[2:0]</th>
<th>SUPP</th>
<th>DA</th>
<th>XXINSEL</th>
<th>XXINBUS[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1E8</td>
<td>ETM_CCER</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 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0 1 1 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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1F0</td>
<td>ETM_TESSEICR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STOPRS[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0x1F8</td>
<td>ETM_TSEVR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FCN[2:0]</td>
<td>RESOURCEB[6:0]</td>
<td>RESOURC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0x200</td>
<td>ETM_TRACEIDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TRACEID[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0x208</td>
<td>ETM_IDR2</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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x314</td>
<td>ETM_PDSR</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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA0</td>
<td>ETM_CLAIMSETR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SET[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>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 1 1 1 1 1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA4</td>
<td>ETM_CLAIMCLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLR[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0xFB0</td>
<td>ETM_LAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>KEY[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0xFB4</td>
<td>ETM_LSR</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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB8</td>
<td>ETM_AUTHSTATR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SNUM[1:0]</td>
<td>SD[0]</td>
<td>NSNUM[1:0]</td>
<td>NSID[1:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0xFC0</td>
<td>ETM_DEVTYPEP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBTYPE[3:0]</td>
<td>MAJORTYP[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0xFD0</td>
<td>ETM_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4KCOUNT[3:0]</td>
<td>JEP106D[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</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 0 0 0 0 0 0 0 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>0xFE0</td>
<td>ETM_PIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 1 0 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>ETM_PIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JEP106ID[3:0]</td>
<td>PARTNUM[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1 0 1 1 1 0 0 0 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</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 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 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>
Refer to Section 41.13: CPU1 ROM table 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  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xFEC  | ETM_PIDR3     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | 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  |
| 0xFF0  | ETM_CIDR0     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFF4  | ETM_CIDR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 0  | 0  | 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  |
| 0xFF8  | ETM_CIDR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFFC  | ETM_CIDR3     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 1  | 0  | 1  | 1  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 41.13: CPU1 ROM table for the register boundary addresses.
41.18 CPU1 trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in Figure 421, the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and four data outputs, TRACEDATA(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes, at the expenses of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

Figure 421. Trace port interface unit (TPIU)

Trace data can also be output on the serial wire output, TRACESWO.

For more information on the Trace Port Interface in the CPU1 refer to the Arm® Cortex®-M4 Technical Reference Manual [2].

41.18.1 TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000
Reset value: 0x0000 000F

<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 PORTSIZE[31:0]: Indicates supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported.

0x0000 000F: Port sizes 1 to 4 supported

### 41.18.2 TPIU current port size register (TPIU_CPSR)

Address offset: 0x004
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>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 PORTSIZE[31:0]: Indicates current trace port size. Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register should only be modified when the formatter is stopped.

### 41.18.3 TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010
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>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 31:13 Reserved, must be kept at reset value.

Bits 12:0 PRESCALER[12:0]: Selects the baud rate for the asynchronous output, TRACESWO. The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

### 41.18.4 TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0
Reset value: 0x0000 0001
41.18.5   **TPIU formatter and flush status register (TPIU_FFSR)**

Address offset: 0x300
Reset value: 0x0000 0008

<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></td>
<td></td>
<td></td>
<td></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>
</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 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: Indicates whether formatter can be stopped or not.
1: Formatter cannot be stopped

Bit 2 **TCPRESENT**: Indicates whether the optional TRACECTL output pin is available for use.
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
0: Formatter has not stopped
1: Formatter has stopped

Bit 0 **FLINPROG**: Flush in progress. Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
0: No flush in progress
1: Flush in progress

41.18.6   **TPIU formatter and flush control register (TPIU_FFCR)**

Address offset: 0x304
Reset value: 0x0000 0102
## Debug support (DBG)

### 41.18.7 TPIU formatter synchronization counter register (TPIU_FSCR)

Address offset: 0x308  
Reset value: 0x0000 0040

<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>
</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></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></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 8** TRIGIN: Trigger on trigger in.  
  1: Indicate a trigger in the trace stream when the TRIGIN input is asserted.

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

- **Bit 1** ENFCONT: Enable continuous formatting. Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output. ETM trace is discarded.  
  0: Continuous formatting is disabled  
  1: Continuous formatting is enabled

- **Bit 0** Reserved, must be kept at reset value.

<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>
</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></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></td>
<td></td>
</tr>
<tr>
<td>Res.</td>
<td>Res.</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CYCCOUNT[12:0]</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>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

- **Bits 12:0** CYCCOUNT[12:0]: Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

### 41.18.8 TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0  
Reset value: 0x0000 000F
41.18.9  **TPIU claim tag clear register (TPIU_CLAIMCLR)**

Address offset: 0xFA4
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 31:4  Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]:** Reset claim tag bits

Write:
- 0000: No effect
- xxx1: Clear bit 0
- xx1x: Clear bit 1
- x1xx: Clear bit 2
- 1xxx: Clear bit 3

Read: Returns current value of claim tag

41.18.10  **TPIU device configuration register (TPIU_DEVIDR)**

Address offset: 0xFC8
Reset value: 0x0000 0CA1
41.18.11 TPIU device type identifier register (TPIU_DEVTYPE)

Address offset: 0xFCC
Reset value: 0x0000 0011

Bits 31:11 Reserved, must be kept at reset value.

Bit 11 **SWONRZ**: Indicates whether Serial wire output, NRZ, is supported.
0x1: Supported

Bit 10 **SWOMAN**: Indicates whether Serial wire output, Manchester encoded format, is supported.
0x1: Supported

Bit 9 **TCLKDATA**: Indicates whether trace clock plus data is supported
0x0: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)
0x1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x1: Two input ports

41.18.12 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0
**Debug support (DBG)**

Reset value: 0x0000 0004

<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>4KCOUNT[3:0]</td>
<td>JEP106CON[3:0]</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>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **4KCOUNT[3:0]**: register file size
- 0x0: Register file occupies a single 4 KB region

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

### 41.18.13 TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 00A1

<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>PARTNUM[7:0]</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>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 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: Part number bits [7:0]
- 0xA1: CPU1 TPIU part number

### 41.18.14 TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B9

<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>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>
**41.18.15 TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)**

Address offset: 0xFE8  
Reset value: 0x0000 004B

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

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications

**41.18.16 TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)**

Address offset: 0xFEC  
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>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 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version  
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified  
0x0: No customer modifications
41.18.17 TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<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
PREAMBLE[7:0]

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

Bits 7:0 PREAMBLE[7:0]: Component ID bits [7:0]

0x0D: Common ID value

41.18.18 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

<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

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

Bits 7:4 CLASS[3:0]: Component ID bits [15:12] - component class
0x9: CoreSight™ component

0x0: Common ID value

41.18.19 TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005
### 41.18.20 TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]

- 0x05: Common ID value

---

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]

- 0xB1: Common ID value
# 41.18.21 CPU1 TPIU register map and reset values

## Table 287. CPU1 TPIU register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TPIU_SSSPR</td>
<td>PORTSIZE[31:0]</td>
</tr>
<tr>
<td>0x04</td>
<td>TPIU_CSPSR</td>
<td>PORTSIZE[31:0]</td>
</tr>
<tr>
<td>0x10</td>
<td>TPIU_ACPR</td>
<td>PRESCALER[12:0]</td>
</tr>
<tr>
<td>0x0F0</td>
<td>TPIU_SPPR</td>
<td></td>
</tr>
<tr>
<td>0x300</td>
<td>TPIU_FFSR</td>
<td></td>
</tr>
<tr>
<td>0x304</td>
<td>TPIU_FFCR</td>
<td></td>
</tr>
<tr>
<td>0x308</td>
<td>TPIU_FSCR</td>
<td>CYCCOUNT[11:0]</td>
</tr>
<tr>
<td>0xFA0</td>
<td>TPIU_CLAIMSETR</td>
<td>CLAIMSET[3:0]</td>
</tr>
<tr>
<td>0xFA4</td>
<td>TPIU_CLAIMCLR</td>
<td>CLAIMCLR[3:0]</td>
</tr>
<tr>
<td>0xFC8</td>
<td>TPIU_DEVIDR</td>
<td>4KCOUNT[3:0]</td>
</tr>
<tr>
<td>0xFD0</td>
<td>TPIU_DEVTYPER</td>
<td>SUBTYPE[3:0]</td>
</tr>
<tr>
<td></td>
<td>TPIU_PIDR4</td>
<td>JEP106CON[3:0]</td>
</tr>
</tbody>
</table>

### Reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TPIU_SSSPR</td>
<td>PORTSIZE[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x04</td>
<td>TPIU_CSPSR</td>
<td>PORTSIZE[31:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x10</td>
<td>TPIU_ACPR</td>
<td>PRESCALER[12:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0F0</td>
<td>TPIU_SPPR</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x300</td>
<td>TPIU_FFSR</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x304</td>
<td>TPIU_FFCR</td>
<td></td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x308</td>
<td>TPIU_FSCR</td>
<td>CYCCOUNT[11:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xFA0</td>
<td>TPIU_CLAIMSETR</td>
<td>CLAIMSET[3:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xFA4</td>
<td>TPIU_CLAIMCLR</td>
<td>CLAIMCLR[3:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xFC8</td>
<td>TPIU_DEVIDR</td>
<td>4KCOUNT[3:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xFD0</td>
<td>TPIU_DEVTYPER</td>
<td>SUBTYPE[3:0]</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>TPIU_PIDR4</td>
<td>JEP106CON[3:0]</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
### Table 287. CPU1 TPIU 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  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xFD4  | TPIU_PIDR5    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD8  | TPIU_PIDR6    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD8  | TPIU_PIDR7    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE0  | TPIU_PIDR0    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE4  | TPIU_PIDR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE8  | TPIU_PIDR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFEC  | TPIU_PIDR3    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF0  | TPIU_CIDR0    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF4  | TPIU_CIDR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF8  | TPIU_CIDR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFC   | TPIU_CIDR3    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
41.19 CPU1 cross trigger interface (CTI)

See Section 41.6: Cross trigger interface (CTI) and matrix (CTM).

41.20 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8 Aug 2013
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B, 10 Dec 2010
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 the characteristics of the microcontroller.

42.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:
- for use as serial number (USB string serial number, or other end applications)
- for use as part of the security keys, 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 memory
- to activate processes such as secure boot.

The 96-bit unique device identifier provides a reference number, unique for a given 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></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 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>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>
</tbody>
</table>

UID[63:48]

UID[47:32]
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

| 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 |

Bits 31:0  **UID[95:64]**: LOT_NUM[55:24]
Lot number (ASCII encoded)

### 42.2 Memory size data register

#### 42.2.1 Flash size data register

Base address: 0x1FFF 75E0
Address offset: 0x00
Read only = 0xXXXX, where X is factory-programmed

| 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 15:0  **FLASH_SIZE[15:0]**: Flash memory size
Indicates the size of the device Flash memory, expressed in Kbytes.
As an example, 0x040 corresponds to 64 Kbytes.

### 42.3 Package data register

Base address: 0x1FFF 7500
Address offset: 0x00
Read only = 0xXXXX, where X is factory-programmed

| 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|     |     |    |     |     |     |     |     |     |     |     | PKG[4:0]|     |     |     |     |     |
| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |
42.4 Part number codification register

Base address: 0x1FFF 77DC
Address offset: 0x00

Read only = 0xXXXX XXXX, where X is factory-programmed

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Bit 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:16]</td>
<td>[15:0]</td>
</tr>
<tr>
<td>CODIFICATION</td>
<td>CODIFICATION</td>
</tr>
</tbody>
</table>

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 PKG[4:0]: Package type
- 10001: WLCSP100 / UFBGA129
- 10011: VFQFPN68
- 01010: UFQFPN48
- Others: reserved

Bits 31:0 CODIFICATION[31:0]: Part number codification
- 0x0000 3533 STMicroelectronics STM32WB35xx part number codification
- 0xXXXX XXXX STMicroelectronics STM32WB55xx part number codification
43 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST’s customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.

Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.

Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.

While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.

All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.
## Revision history

### Table 288. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-Jan-2018</td>
<td>1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
| 12-Apr-2018| 2        | Updated document title, *Introduction* and *Related documents*. Updated *Figure 1: System architecture* and *Figure 2: Memory map*. Updated *Table 1: Memory map and peripheral register boundary addresses*. Updated Section 2.1.4: S3: CPU2 (Cortex®-M0+) S-bus, CPU1 physical remap and Section 2.33: CPU2 boot, and added Section 2.34: CPU2 SRAM fetch disable. Updated Section 3.3.4: Read access latency, Section 3.3.8: Flash main memory programming sequences, Standard programming, Fast programming, Programming errors signaled by flags, PGSERR and PGAERR in a page-based row programming, User and read protection option bytes, Secure SRAM2 start address and CPU2 reset vector option bytes, Option byte loading, Section 3.5: FLASH UID64, Changing the Read protection level, Section 3.6.4: CPU2 security (ESE), CPU2 secure SRAM2 areas, Section 3.10.1: Flash memory access control register (FLASH_ACR), Section 3.10.4: Flash memory status register (FLASH_SR), Section 3.10.5: Flash memory control register (FLASH_CR), Section 3.10.7: Flash memory option register (FLASH_OPTR), Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR) and Section 3.10.16: Flash memory CPU2 status register (FLASH_C2SR). Added Programming errors causing a bus error and PGSERR and PGAERR in a page-based row programming. Updated *Table 4: Number of wait states vs, Flash memory clock (HCLK4) frequency*, Table 9: Option bytes organization, Table 13: RDP regression from Level 1 to Level 0 and memory erase and Table 18: Flash interface register map and reset values. Updated Section 5.4.2: CRC independent data register (CRC_IDR) and Section Table 20.: CRC register map and reset values. Updated Section 6.1: Power supplies, Section 6.4: Low-power modes, Section : Entering Low-power run mode, Section 6.6.3: PWR control register 3 (PWR_CR3), Section 6.6.5: PWR status register 1 (PWR_SR1), Section 6.6.7: PWR status clear register (PWR_SCR), Section 6.6.8: PWR control register 5 (PWR_CR5), Section 6.6.21: PWR CPU2 control register 1 (PWR_C2CR1) and Section 6.6.22: PWR CPU2 control register 3 (PWR_C2CR3). Updated Table 24: Low-power mode summary, Table 30: Stop0 mode, Table 31: Stop1 mode, Table 32: Stop2 mode, Table 33: Standby mode, Table 34: Shutdown mode and Table 35: PWR register map and reset values. Updated *Figure 7: Power supply overview* and *Figure 8: Supply configurations*. Updated Table 37: Peripherals interconnect matrix and Section 7.4.1: From timer (TIM1/TIM2/TIM17) to timer (TIM1/TIM2).
## Table 288. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-Apr-2018</td>
<td>2</td>
<td>Updated Figure 15: Clock tree. Updated Table 38: Maximum clock source frequency, Table 39: SMPS step-down converter clock source selection and division and Table 42: RCC register map and reset values. Updated Section 8.2.5: PLLs, Section 8.2.8: LSI2 clock, Section 8.2.9: System clock (SYSCLK) selection, Section 8.2.13: LSI source selection, Section 8.2.20: Clock-out capability, Section 8.2.22: Peripheral clocks enable, Section 8.3: Low-power modes, Section 8.4.1: RCC clock control register (RCC_CR), Section 8.4.9: RCC SMPS step-down converter control register (RCC_SMPS), Section 8.4.16: RCC APB3 peripheral reset register (RCC_APB3RSTR), Section 8.4.19: RCC AHB3 and AHB4 peripheral clock enable register (RCC_AHB3ENR), Section 8.4.22: RCC APB2 peripheral clock enable register (RCC_APB2ENR), Section 8.4.29: RCC peripherals independent clock configuration register (RCC_CCIPR), Section 8.4.31: RCC control/status register (RCC_CSR), Section 8.4.34: RCC extended clock recovery register (RCC_EXTCFGR), Section 8.4.35: RCC CPU2 AHB1 peripheral clock enable register (RCC_C2AHB1ENR), Section 8.4.37: RCC CPU2 AHB3 and AHB4 peripheral clock enable register (RCC_C2AHB3ENR), Section 8.4.41: RCC CPU2 APB3 peripheral clock enable register (RCC_C2APB3ENR), Section 8.4.42: RCC CPU2 AHB1 peripheral clocks enable in Sleep modes register (RCC_C2AHB1SMENR) and Section 8.4.48: RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR). Updated Section 8.4.1: General-purpose I/O (GPIO), Section 8.4.2: I/O pin alternate function multiplexer and mapping, Section 8.5.1: GPIO port mode register (GPIOx_MODE) (x =A to I) and sections from 8.5.4 to 8.4.11. Updated Section 10.1: SYSCFG main features, Section 10.2.7: SYSCFG SRAM2 control and status register (SYSCFG_SCSR), Section 10.2.16: SYSCFG secure IP control register (SYSCFG_SIPCR) and Table 46: SYSCFG register map and reset values. Updated Table 60: CPU1 vector table, Table 61: CPU2 vector table and Table 62: Wakeup interrupt table. Updated title of Section 14: Extended interrupt and event controller (EXTI), Section 14.4: EXTI functional description, sections 14.6.1 to 14.6.8, and replaced former sections 12.5.9 to 12.5.12 with new sections 14.6.9 to 14.6.16. Updated Table 69: EXTI register map and reset values and Table 62: Wakeup interrupt table. Removed former sections 14.5.13 to 14.5.18. Updated Section 15.1: Introduction, Section 15.2: QUADSPI main features, Section 15.3.2: QUADSPI pins, Section 15.3.8: QUADSPI flash memory configuration, Section 15.3.10: QUADSPI configuration, Section 15.3.15: NCS behavior, Section 15.5.1: QUADSPI control register (QUADSPI_CR) and Section 15.5.6: QUADSPI communication configuration register (QUADSPI_CCR) Updated Table 70: QUADSPI pins and Table 72: QUADSPI register map and reset values. Removed former Dual-flash mode and former sections 23.7.5 to 23.7.7.</td>
</tr>
</tbody>
</table>
Updated Section 16.7.5: ADC configuration register 2 (ADC_CFR2).
Updated Figure 40: ADC block diagram and Figure 48: Stopping ongoing regular conversions.
Updated RSA encryption/decryption principle and Table 157: PKA register map and reset values.
Updated Section 24.3.29: Debug mode.
Updated notes in Using one timer to start another timer and in TIM2 slave mode control register (TIM2_SMCR).
Updated Section 31.3.4: How to program the watchdog timeout, Section 31.5.2: WWDG configuration register (WWDG_CFR) and Table 185: WWDG register map and reset values.
Updated Selecting the clock source and the appropriate oversampling method, Section 34.7.4: LPUART control register 3 (LPUART_CR3), sections 34.7.7 to 34.7.12 and Table 178: USART interrupt requests.
Updated Frame length, Section 36.4.8: SAI clock generator and its subsections, Wrong clock configuration in master mode (with NODIV = 0), and made two different sections for similar registers in Section 36.6: SAI registers.
Updated Table 230: Clock generator programming examples and Table 236: SAI interrupt sources.
Updated Table 240: IPCC register map and reset values.
Updated Figure 410: CRS block diagram.
Updated Table 1: Memory map and peripheral register boundary addresses.
Updated Secure system flash memory programming, Section 3.3.7: Flash main memory erase sequences, Flash memory mass erase, Section 3.3.8: Flash main memory programming sequences, User and read protection option bytes, Secure flash memory start address option bytes, Secure SRAM2 start address and CPU2 reset vector option bytes, Secure user options, WRP Area A address option bytes, WRP Area B address option bytes, Changing the CPU2 security mode, Level 2: No debug, Changing the Read protection level, Section 3.6.4: CPU2 security (ESE), CPU2 secure SRAM2 areas, CPU2 debug access, Section 3.10.5: Flash memory control register (FLASH_CR), Section 3.10.7: Flash memory option register (FLASH_OPTR), Section 3.10.18: Secure flash memory start address register (FLASH_SFR), Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR) and Section 3.10.17: Flash memory CPU2 control register (FLASH_C2CR).
Added Section 3.9: Register access protection.
Updated Table 3: Flash memory - Single bank organization, Table 5: Page erase overview, Table 6: Mass erase overview, Table 8: Option bytes format, Table 9: Option bytes organization, Table 13: RDP regression from Level 1 to Level 0 and memory erase and Table 18: Flash interface register map and reset values.

Updated titles of sections 5.4.1 to 5.4.5.

Updated Section 6.1.1: Independent analog peripherals supply, Section 6.4: Low-power modes, Section 6.6.3: PWR control register 3 (PWR_CR3), Section 6.6.5: PWR status register 1 (PWR_SR1), sections 6.6.8 to 6.6.20, 6.6.22 and 6.6.23.

Updated Table 23: Sub-system low power wake-up sources, Table 24: Low-power mode summary and Table 25: Functionalities depending on system operating mode.

Updated Table 37: Peripherals interconnect matrix.

Updated Section 8.2: Clocks, Section 8.2.8: LSI2 clock, LSI2 trimming parameter, Section 8.2.20: Clock-out capability, Section 8.3: Low-power modes, Section 8.4.1: RCC clock control register (RCC_CR) and Section 8.4.31: RCC control/status register (RCC_CSR).

Updated Table 40: Peripheral clock enable, Table 42: RCC register map and reset values and added Table 41: Single core Low power debug configurations.

Updated Section 10.2.16: SYSCFG secure IP control register (SYSCFG_SIPCR).

Updated Section 12.4.3: DMAMUX channels, Section 12.4.3: DMAMUX channels, Synchronization overrun and interrupt, Trigger overrun and interrupt, Section 12.6.1: DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR), Section 12.6.3: DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) and Section 12.6.6: DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR).

Updated Figure 26: DMAMUX block diagram.

Replaced AIEC with EXTI in Section 13: Nested vectored interrupt controller (NVIC). and ID with CoreID in Section 38: Hardware semaphore (HSEM).

Updated Table 60: CPU1 vector table and Table 61: CPU2 vector table.

Updated Section 15.2: QUADSPI main features, FIFO and data management, Section 15.3.10: QUADSPI configuration, Section 15.3.11: QUADSPI use, Section 15.5.1: QUADSPI control register (QUADSPI_CR), Section 15.5.3: QUADSPI status register (QUADSPI_SR) and Section 15.5.6: QUADSPI communication configuration register (QUADSPI_CCR).

Updated Table 72: QUADSPI register map and reset values.
### Table 288. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>08-Jan-2019</td>
<td>3</td>
<td>Updated Section 16.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN), Section 16.4.7: Single-ended and differential input channels, Section 16.4.8: Calibration (ADCAL, ADCALDF, ADC_CALFACT), Section 16.4.10: Constraints when writing the ADC control bits, Section 16.4.11: Channel selection (SQRx, JSQRx) and Section 16.4.12: Channel-wise programmable sampling time (SMPR1, SMPR2). Updated Table 74: ADC internal input/output signals and Figure 42: ADC1 connectivity. Updated Section 21.1: Introduction, Section 21.2: RNG main features and Health checks. Updated Montgomery space and fast mode operations, RSA encryption/decryption principle, Executing a PKA operation, Using precomputed Montgomery parameters (PKA fast mode), Section 23.3.7: PKA error management, Extended ECDSA support, Section 23.5.1: Supported elliptic curves, Section 23.7.1: PKA control register (PKA_CR), Section 23.7.2: PKA status register (PKA_SR), Section 23.7.3: PKA clear flag register (PKA_CLRFR) and Section 23.7.4: PKA RAM. Updated Table 129: Modular addition, Table 130: Modular subtraction, Table 131: Montgomery multiplication, Table 135: Modular reduction, Table 136: Arithmetic addition, Table 137: Arithmetic subtraction, Table 138: Arithmetic multiplication, Table 139: Arithmetic comparison, Table 175: ECC curves parameters, Table 150: Modular exponentiation computation times, Table 151: ECC scalar multiplication computation times, Table 152: ECDSA signature average computation times, Table 153: ECDSA verification average computation times and Table 157: PKA register map and reset values. Replaced BKEx, BKPx by BKE, BK2E, BK2P and split registers CCMR1 and CCMR2 in Section 24: Advanced-control timer (TIM1). Updated Section 24.3.16: Using the break function. Updated Figure 144: Advanced-control timer block diagram and Figure 191: Output redirection (BRK2 request not represented). Updated Section 26.3.19: Debug mode and Section 26.4.6: TIMx capture/compare mode register 1 (TIMx_CCMRx)(x = 16 to 17). Updated Table 170: TIM16/TIM17 register map and reset values. Updated Figure 275: Output redirection. Updated Section 29.2: RTC main features and Section 29.7.20: RTC backup registers (RTC_BKPxR). Updated Table 183: RTC register map and reset values. Added Section 30.3.4: Low-power freeze. Updated Section 49.3.1: IWDG block diagram and Section 30.4.2: IWDG prescaler register (IWDG_PR). Removed former Section 49.3.6: Behavior in Stop and Standby modes. Updated Table 184: IWDG register map and reset values. Updated Section 38.3.8: AHB bus master ID verification and Sections 38.4.1 to 32.4.7. Added Table 242: Authorized AHB bus master IDs and updated Table 243: HSEM register map and reset values.</td>
</tr>
</tbody>
</table>
Table 288. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>08-Jan-2019</td>
<td>3</td>
<td>Updated Section 41.1: Introduction, Section 41.4.8: DP target identification register (DP_TARGETIDR), Section 41.7: Cross trigger interface registers, Section 41.8.1: DBGMCU identity code register (DBGMCU_IDCODE), Section 41.10.14: DWT CoreSight peripheral identity register 1 (DWT_PIDR1) and Section 41.14.14: DWT CoreSight peripheral identity register 1 (DWT_PIDR1). Updated Table 267: Debug port register map and reset values and Table 274: DBGMCU register map and reset values. Removed former Section 32.11: CPU2 Instrumentation Trace Macrocell (ITM) and its subsections. Updated Section 42.1: Unique device ID register (96 bits)</td>
</tr>
<tr>
<td>01-Mar-2019</td>
<td>4</td>
<td>Changed document classification, from ST restricted to Public. Updated Section 8.3: Low-power modes, Channel configuration procedure, Channel state and disabling a channel, Section 21.1: Introduction, Section 21.2: RNG main features, Section 21.3.3: Random number generation and its subsections, Section 21.3.4: RNG initialization, Section 21.3.5: RNG operation and its subsections, Section 21.3.6: RNG clocking, Section 21.3.7: Error management, Section 21.3.8: RNG low-power use, Section 21.6.1: Introduction, Section 21.7: RNG registers, Section 21.7.2: RNG status register (RNG_SR), Section 21.7.3: RNG data register (RNG_DR), sections 25.4.7 to 25.4.24, 26.4.6 and 26.4.7, Section 33.5.4: USART FIFOs and thresholds, Section 34.4.4: LPUART FIFOs and thresholds and Section 50.4.14: SPDIF output. Updated Table 62: Wakeup interrupt table, Table 167: TIM2 register map and reset values and Table 260: STM32WB55xx SPI implementation. Updated Figure 113: RNG block diagram, Figure 114: Entropy source model and Figure 355: Reception using DMA. Added Figure 115: RNG initialization overview and footnote to Figure 370: Packing data in FIFO for transmission and reception.</td>
</tr>
<tr>
<td>13-Feb-2020</td>
<td>5</td>
<td>Updated Section 2.1: System architecture, Section 3.5: FLASH UID64, Section 3.6.4: CPU2 security (ESE), Section 6.4.12: Auto wake-up from Low-power mode, Section 6.6.8: PWR control register 5 (PWR_CRS), Section 7.4.7: From USB to timer (TIM2), Software reset, Section 8.2: Clocks, External source, Section 8.2.2: HSI16 clock, Section 8.2.3: MSI clock, Section 8.2.5: PLLs, Section 8.2.8: LSI2 clock, Section 8.4.1: RCC clock control register (RCC_CR), Section 8.4.4: RCC PLL configuration register (RCC_PLLCFGR), Section 8.4.31: RCC control/status register (RCC_CSR), Section 8.4: GPIO functional description, Section 8.4.2: I/O pin alternate function multiplexer and mapping, Section 8.4.7: I/O alternate function input/output, Section 10.2.15: SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2), Section 13.1: NVIC main features, Section 16.4.32: Monitoring the internal voltage reference, Section 29.5: RTC low-power modes, Section 36.4.10: PDM interface, Section 40.7.2: CRS configuration register (CRS_CFRG) and Section 42.3: Package data register. Added note in Section 3.8: FLASH interrupts and Section 40.3: CRS implementation. Removed note from Section 19.2: LCD main features.</td>
</tr>
<tr>
<td>Date</td>
<td>Revision</td>
<td>Changes</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>13-Feb-2020</td>
<td>5 (cont'd)</td>
<td>Updated Table 1: Memory map and peripheral register boundary addresses, Table 2: Boot modes, Table 25: Functionalities depending on system operating mode, Table 42: RCC register map and reset values, Table 60: CPU1 vector table, Table 61: CPU2 vector table and Table 62: Wakeup interrupt table. Updated Figure 15: Clock tree, Figure 19: Three-volt or Five-volt tolerant GPIO structure (TT or FT), Figure 25: Input floating / pull up / pull down configurations, Figure 26: Output configuration and Figure 27: Alternate function configuration, and removed former Figure 21: Basic structure of a 5-Volt tolerant I/O port bit.</td>
</tr>
<tr>
<td>11-Jul-2020</td>
<td>6</td>
<td>Introduced STM32WB35xx devices. Updated Introduction, Section 3.3.1: Flash memory organization, PCROP1A start address option bytes, PCROP1A end address option bytes, WRP Area A address option bytes, WRP Area B address option bytes, PCROP1B start address option bytes, PCROP1B end address option bytes, Section 3.5: FLASH UID64, Section 3.10.5: Flash memory control register (FLASH_CR), sections 3.10.8 to 3.10.13, Section 3.10.17: Flash memory CPU2 control register (FLASH_C2CR), sections 6.6.2 to 6.6.4, Section 6.6.7: PWR status clear register (PWR_SCR), sections 6.6.12 to 6.6.14 and 6.6.18 to 6.6.20, Section 6.6.22: PWR CPU2 control register 3 (PWR_C2CR3), Section 7.4.8: From internal analog to ADC1, sections 8.4.12 to 8.4.13, 8.4.17 to 8.4.18, Section 8.4.20: RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1), sections 8.4.23 to 8.4.24, Section 8.4.26: RCC APB1 peripheral clocks enable in Sleep mode register 1 (RCC_APB1SMENR1), Section 8.4.45: RCC CPU2 APB1 peripheral clocks enable in Sleep mode register 1 (RCC_C2APB1SMENR1), Section 8.4.48: RCC CPU2 APB3 peripheral clock enable in Sleep mode register (RCC_C2APB3SMENR), Section 9.4.3: I/O port control registers, sections 10.2.3 to 10.2.7, Section 10.2.15: SYSCFG CPU2 interrupt mask register 2 (SYSCFG_C2IMR2), Section 12.4.4: DMAMUX request line multiplexer, Calculating the actual VREF+ voltage using the internal reference voltage, Section 17.5.2: VREFBUF calibration control register (VREFBUF_CCR), Section 18.6.1: Comparator 1 control and status register (COMP1_CSR), Section 20.2: TSC main features, Section 20.3.4: Charge transfer acquisition sequence, Montgomery space and fast mode operations, Section 23.7.2: PKA status register (PKA_SR), Section 16.4.7: Single-ended and differential input channels, Software procedure to enable the ADC, Section 24.4.1: TIM1 control register 1 (TIM1_CR1), Section 24.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1), Section 24.4.20: TIM1 break and dead-time register (TIM1_BDTR), sections 24.4.27 to 24.4.28, Section 25.3.19: Timer synchronization, Section 25.4.8: TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1), sections 27.7.1 to 27.7.2, Section 34.7.8: LPUART interrupt and status register [alternate] (LPUART_ISR), Section 36.6.17: SAI PDM control register (SAI_PDMCR), Section 41.1: Introduction, Section 41.8.2: DBGMCU configuration register (DBGMCU_CR), Section 41.8.4: DBGMCU CPU2 APB1 peripheral freeze register 1 (DBGMCU_C2APB1FZR1), Section 41.8.6: DBGMCU CPU2 APB1 peripheral freeze register 2 (DBGMCU_C2APB1FZR2) and Section 41.8.8: DBGMCU CPU2 APB2 peripheral freeze register (DBGMCU_C2APB2FZR).</td>
</tr>
</tbody>
</table>
### Table 288. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>11-Jul-2020</td>
<td>6 (cont'd)</td>
<td>Updated Table 3: Flash memory - Single bank organization, Table 94: COMP1 input plus assignment, Table 95: COMP1 input minus assignment, Table 150: Modular exponentiation computation times, Table 155: Montgomery parameters average computation times, Table 162: Output control bits for complementary OCx and OCxN channels with break feature, Table 166: Output control bit for standard OCx channels and Table 169: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17). Updated Figure 167: Control circuit in normal mode, internal clock divided by 1, Figure 173: Capture/compare channel 1 main circuit, Figure 204: General-purpose timer block diagram, Figure 230: Capture/Compare channel 1 main circuit, Figure 253: TIM16/TIM17 block diagram and Figure 267: Capture/compare channel 1 main circuit. Added Section 7.2: Interconnect matrix implementation, Note: to Section 8.4.3: RCC clock configuration register (RCC_CFGR), Section 9.3: GPIO implementation, Section 13.2: NVIC implementation, Section 14.2: EXTI implementation, Section 16.3: ADC implementation, Section 17.2: VREFBUF implementation, Section 26.3.18: Using timer output as trigger for other timers (TIM16/TIM17), Section 29.3: RTC implementation and Section 36.3: SAI implementation. Added Table 51: DMAMUX implementation, Table 154: Point on elliptic curve Fp check average computation times, Table 147: LPTIM implementation, Table 186: I2C implementation and footnote to Table 223: SPI implementation. Minor text edits across the whole document.</td>
</tr>
<tr>
<td>20-Apr-2021</td>
<td>7</td>
<td>Updated Figure 109: Surface charge transfer analog I/O group structure, Figure 343: RS232 RTS flow control, Figure 344: RS232 CTS flow control, Figure 385: Detailed PDM interface block diagram and Figure 400: IPCC Simplex - Send procedure state diagram. Updated Table 106: Acquisition sequence summary, Table 149: Family of supported curves for ECC operations, Table 156: PKA interrupt requests, Table 216: Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32.768 kHz, Table 217: Error calculation for programmed baud rates at fCK = 100 MHz and Table 241: HSEM internal input/output signals. Added Table 238: IPCC interface signals. Added footnotes to Table 131: Montgomery multiplication, Table 226: STM32WB55xx SAI features, Table 228: SAI input/output pins and to Figure 377: SAI functional block diagram, Figure 385: Detailed PDM interface block diagram. Added Section 33.6: USART in low-power modes and Section 34.5: LPUART in low-power modes. Added notes in Section 27.4.6: Trigger multiplexer and Section 20.3.2: Surface charge transfer acquisition overview.</td>
</tr>
</tbody>
</table>
### Table 288. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-Apr-2021</td>
<td>7 (cont’d)</td>
<td>Updated <em>Introduction</em>, <em>Related documents</em>, ST production values in Section 3.4.1: Option bytes description, Section 3.10.6: Flash memory ECC register (FLASH_ECCR), Section 3.10.9: Flash memory PCROP zone A end address register (FLASH_PCROP1AER), Section 3.10.10: Flash memory WRP area A address register (FLASH_WRP1AR), Section 3.10.11: Flash memory WRP area B address register (FLASH_WRP1BR), Section 3.10.12: Flash memory PCROP zone B start address register (FLASH_PCROP1BSR), Section 3.10.13: Flash memory PCROP zone B end address register (FLASH_PCROP1BER), Section 3.10.14: Flash memory IPCC mailbox data buffer address register (FLASH_IPCCBR), Section 3.10.18: Secure flash memory start address register (FLASH_SFR), Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR), Section 4.1: Introduction, Section 4.2: Main features, Section 5.2: CRC main features, Polynomial programmability, Section 5.4: CRC registers, Section 8.2: Clocks, Section 8.2.14: SMPS step-down converter clock, Section 8.4.1: RCC clock control register (RCC_CR), Section 8.4.3: RCC clock configuration register (RCC_CFGR), Section 16.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN), note in Triggered injection mode, Section 20.2: TSC main features, Section 23.3.4: PKA public key acceleration, Section 23.5.1: Supported elliptic curves, Section 36.4.10: PDM interface, Section 27.7.1: LPTIM interrupt and status register (LPTIM_ISR), Section 33.2: USART main features, Section 33.8.9: USART interrupt and status register (USART_ISR), Section 33.8.10: USART interrupt and status register [alternate] (USART_ISR), Section 34.4.14: LPUART low-power management, Section 36.4.10: PDM interface, Section 36.6.18: SAI PDM delay register (SAI_PDMDLY), Section 38.3.3: HSEM lock procedures, Section 38.3.5: HSEM unlock procedures, Section 38.3.6: HSEM COREID semaphore clear, Section 38.3.7: HSEM interrupts, Section 38.3.8: AHB bus master ID verification, Section 41.4.7: DP data link control register (DP_DLCR), Simplex communications, Section 41.5.1: AP control/status word register (AP_CSWR), Section 41.5.5: AP base address register (AP_BASER), Section 41.5.6: AP identification register (AP_IDR) and Section 41.8.2: DBGMCU configuration register (DBGMCU_CR).</td>
</tr>
</tbody>
</table>

Minor text edits across the whole document.
<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 13-Aug-2021| 8        | Updated Section 3.5: FLASH UID64, Section 5.2: CRC main features, Section 6.6.2: PWR control register 2 (PWR_CR2), Section 8.2.8: LSI2 clock, Section 8.2.12: Clock security system on LSE (LSECSS), Section 9.4.2: I/O pin alternate function multiplexer and mapping, Section 24.3.16: Using the break function, Section 24.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1), Section 25.4.8: TIM2 capture/compare mode register 1 [alternate] (TIM2_CCMR1), Section 26.3.11: Using the break function, Section 29.4.14: Calibration clock output, Section 33.8.3: USART control register 2 (USART_CR2) and Section 41.8.1: DBGMCU identity code register (DBGMCU_IDCODE).  
Added Section 26.3.13: 6-step PWM generation.  
Changed SCLK into CK throughout Section 33: Universal synchronous/asynchronous receiver transmitter (USART/UART).  
Minor text edits across the whole document.  
Added footnote 2 to Table 21: Supply configuration control.  
Updated Figure 232: TDM frame configuration examples and Table 274: DBGMCU register map and reset values.  
Updated Figure 144: Advanced-control timer block diagram, Figure 167: Control circuit in normal mode, internal clock divided by 1, Figure 263: Control circuit in normal mode, internal clock divided by 1, Figure 323: TC/TXE behavior when transmitting, Figure 324: Start bit detection when oversampling by 16 or 8, Figure 332: USART example of synchronous master transmission and footnotes of Figure 385: Detailed PDM interface block diagram. |
| 04-Oct-2021| 9        | Updated Section 3.5: FLASH UID64, DMA operation in different operating modes, Section 33.5.20: RS232 hardware flow control and RS485 Driver Enable, Section 33.8.4: USART control register 3 (USART_CR3), Section 34.4.13: RS232 hardware flow control and RS485 Driver Enable, Section 34.7.4: LPUART control register 3 (LPUART_CR3) and Section 41.8.1: DBGMCU identity code register (DBGMCU_IDCODE).  
Updated Figure 135: GCM authenticated encryption.  
Added Section 42.4: Part number codification register.  
Minor text edits across the whole document. |
Table 288. Document revision history (continued)

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>07-Jun-2022</td>
<td>10</td>
<td>Updated Introduction, Section 3.3.1: Flash memory organization, Section 3.4.1: Option bytes description, Section 4.1: Introduction, Section 4.2: Main features, Section 6.4.4: Exiting Low-power mode, Section 8.4.5: RCC PLLSAI1 configuration register (RCC_PLLSAI1CFGR), Section 8.4.30: RCC backup domain control register (RCC_BDCR), Section 8.4.31: RCC control/status register (RCC_CSR), Section 8.4.33: RCC clock HSE register (RCC_HSECR), Analog watchdog, Triggering the start of a command, Section 16.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN), Reading the temperature, Section 21.2: RNG main features, Section 41.7.3: CTI application trigger set register (CTI_APPSETR), Section 41.7.15: CTI lock access register (CTI_LAR), and Section 41.8.2: DBGMCU configuration register (DBGMCU_CR). Added Section 16.5: ADC in low-power mode and Section 43: Important security notice. Updated Figure 57: Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs during an ongoing conversion, Figure 59: Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) Case when JADSTP occurs outside an ongoing conversion, Figure 87: Triggered regular oversampling with injection, Figure 135: GCM authenticated encryption, Figure 299: Transfer bus diagrams for I2C slave transmitter (mandatory events only), Figure 302: Transfer bus diagrams for I2C slave receiver (mandatory events only), and Figure 309: Transfer bus diagrams for I2C master transmitter (mandatory events only). Updated Table 1: ADC input/output pins and Table 225: SPI register map and reset values. Minor text edits across the whole document.</td>
</tr>
<tr>
<td>01-Feb-2023</td>
<td>11</td>
<td>Added Empty check and Caution in Section 3.3.6: Flash memory program and erase operations. Updated Section 3.3.7: Flash main memory erase sequences, Section 3.3.8: Flash main memory programming sequences, note in Section 3.6.1: Read protection (RDP), Section 3.10.4: Flash memory status register (FLASH_SR), Section 3.10.5: Flash memory control register (FLASH_CR), Section 3.10.19: Flash memory secure SRAM2 start address and CPU2 reset vector register (FLASH_SRRVR), Section 6.1: Power supplies, Force SMPS step-down converter Bypass mode, Section 6.2.2: Programmable voltage detector (PVD), Section 8.1.2: System reset, Section 8.2.14: SMPS step-down converter clock, Converting a supply-relative ADC measurement to an absolute voltage value, Section 1.12.4: ADC configuration register 1 (ADC_CFR1), Section 24.4.7: TIM1 capture/compare mode register 1 (TIM1_CCMR1), Section 25.4.7: TIM2 capture/compare mode register 1 (TIM2_CCMR1), Section 31.4: WWDG interrupts, Section 31.5.2: WWDG configuration register (WWDG_CFR), and Section 41.4.11: DP access port select register (DP_SELECTR). Updated Figure 109: Surface charge transfer analog I/O group structure, Figure 110: Sampling capacitor voltage variation, and Figure 288: Watchdog block diagram. Updated Table 267: Debug port register map and reset values. Minor text edits across the whole document.</td>
</tr>
<tr>
<td>Date</td>
<td>Revision</td>
<td>Changes</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 04-Sep-2023| 12       | Updated **Introduction, Section 3.5: FLASH UID64, Section 3.10.16: Flash memory CPU2 status register (FLASH_C2SR), Section 4.1: Introduction, Section 4.2: Main features, Entering Stop0 mode, Section 6.4.8: Stop1 mode, Entering Stop2 mode, Section 8.2.8: LSI2 clock, Section 8.2.20: Clock-out capability, Section 20.3.4: Charge transfer acquisition sequence, Section 18.6.1: Comparator 1 control and status register (COMP1_CSR), Section 24.3.22: Encoder interface mode, and Section 25.3.15: Encoder interface mode.**  
Updated **Figure 2: Memory map, Figure 287: Independent watchdog block diagram, Figure 293: I2C initialization flow, and Figure 296: Slave initialization flow.**  
Minor text edits across the whole document. |
| 04-Jan-2024| 13       | Updated **Section 6.4.12: Auto wake-up from Low-power mode.**  
Added **Section 32.4.15: SMBus master mode** and removed former **Section 32.4.17: DMA requests and Section 32.4.18: Debug mode.**  
Added **Table 204: USART/UART input/output pins, Table 205: USART internal input/output signals, Table 214: LPUART input/output pins, and Table 215: LPUART internal input/output signals.**  
Updated **Figure 347: LPUART block diagram.**  
Reorganized register sections by address offset in **Section 36: Serial audio interface (SAI).**  
Minor text edits across the whole document. |
# Index

## A

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_AWD2CR</td>
<td>502</td>
</tr>
<tr>
<td>ADC_AWD3CR</td>
<td>503</td>
</tr>
<tr>
<td>ADC_CALFACT</td>
<td>504</td>
</tr>
<tr>
<td>ADC_CCR</td>
<td>505</td>
</tr>
<tr>
<td>ADC_CFGR</td>
<td>486</td>
</tr>
<tr>
<td>ADC_CFG2R</td>
<td>490</td>
</tr>
<tr>
<td>ADC_CR</td>
<td>483</td>
</tr>
<tr>
<td>ADC_CSR</td>
<td>504</td>
</tr>
<tr>
<td>ADC_DIFSEL</td>
<td>503</td>
</tr>
<tr>
<td>ADC_DR</td>
<td>498</td>
</tr>
<tr>
<td>ADC_IER</td>
<td>481</td>
</tr>
<tr>
<td>ADC_ISR</td>
<td>479</td>
</tr>
<tr>
<td>ADC_JDRy</td>
<td>502</td>
</tr>
<tr>
<td>ADC_JSQR</td>
<td>499</td>
</tr>
<tr>
<td>ADC_OFrY</td>
<td>501</td>
</tr>
<tr>
<td>ADC_SMPR1</td>
<td>491</td>
</tr>
<tr>
<td>ADC_SMPR2</td>
<td>492</td>
</tr>
<tr>
<td>ADC_SQR1</td>
<td>495</td>
</tr>
<tr>
<td>ADC_SQR2</td>
<td>496</td>
</tr>
<tr>
<td>ADC_SQR3</td>
<td>497</td>
</tr>
<tr>
<td>ADC_SQR4</td>
<td>498</td>
</tr>
<tr>
<td>ADC_TR1</td>
<td>493</td>
</tr>
<tr>
<td>ADC_TR2</td>
<td>493</td>
</tr>
<tr>
<td>ADC_TR3</td>
<td>494</td>
</tr>
<tr>
<td>AES_CR</td>
<td>626</td>
</tr>
<tr>
<td>AES_DINr</td>
<td>630</td>
</tr>
<tr>
<td>AES_DOUTR</td>
<td>630</td>
</tr>
<tr>
<td>AES_IVR0</td>
<td>633</td>
</tr>
<tr>
<td>AES_IVR1</td>
<td>633</td>
</tr>
<tr>
<td>AES_IVR2</td>
<td>633</td>
</tr>
<tr>
<td>AES_IVR3</td>
<td>634</td>
</tr>
<tr>
<td>AES_KEYR0</td>
<td>631</td>
</tr>
<tr>
<td>AES_KEYR1</td>
<td>632</td>
</tr>
<tr>
<td>AES_KEYR2</td>
<td>632</td>
</tr>
<tr>
<td>AES_KEYR3</td>
<td>632</td>
</tr>
<tr>
<td>AES_KEYR4</td>
<td>634</td>
</tr>
<tr>
<td>AES_KEYR5</td>
<td>634</td>
</tr>
<tr>
<td>AES_KEYR6</td>
<td>635</td>
</tr>
<tr>
<td>AES_KEYR7</td>
<td>635</td>
</tr>
<tr>
<td>AES_SR</td>
<td>629</td>
</tr>
<tr>
<td>AES_SUSPxR</td>
<td>635</td>
</tr>
<tr>
<td>AP_BD0-3R</td>
<td>1369</td>
</tr>
<tr>
<td>AP_DWR</td>
<td>1369</td>
</tr>
<tr>
<td>AP_TAR</td>
<td>1369</td>
</tr>
</tbody>
</table>

## B

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPU_CIDR0</td>
<td>1433</td>
</tr>
<tr>
<td>BPU_CIDR1</td>
<td>1434</td>
</tr>
<tr>
<td>BPU_CIDR2</td>
<td>1434</td>
</tr>
<tr>
<td>BPU_CIDR3</td>
<td>1435</td>
</tr>
<tr>
<td>BPU_COMPxR</td>
<td>1431</td>
</tr>
<tr>
<td>BPU_CTRLR</td>
<td>1430</td>
</tr>
<tr>
<td>BPU_PIDR0</td>
<td>1432</td>
</tr>
<tr>
<td>BPU_PIDR1</td>
<td>1432</td>
</tr>
<tr>
<td>BPU_PIDR2</td>
<td>1432</td>
</tr>
<tr>
<td>BPU_PIDR3</td>
<td>1433</td>
</tr>
<tr>
<td>BPU_PIDR4</td>
<td>1431</td>
</tr>
<tr>
<td>BPU_REMAPR</td>
<td>1430</td>
</tr>
</tbody>
</table>

## C

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1ROM_CIDR0</td>
<td>1441</td>
</tr>
<tr>
<td>C1ROM_CIDR1</td>
<td>1441</td>
</tr>
<tr>
<td>C1ROM_CIDR2</td>
<td>1442</td>
</tr>
<tr>
<td>C1ROM_CIDR3</td>
<td>1444</td>
</tr>
<tr>
<td>C1ROM_MEMTYPEr</td>
<td>1438</td>
</tr>
<tr>
<td>C1ROM_PIDR0</td>
<td>1439</td>
</tr>
<tr>
<td>C1ROM_PIDR1</td>
<td>1440</td>
</tr>
<tr>
<td>C1ROM_PIDR2</td>
<td>1440</td>
</tr>
<tr>
<td>C1ROM_PIDR3</td>
<td>1441</td>
</tr>
<tr>
<td>C1ROM_PIDR4</td>
<td>1439</td>
</tr>
<tr>
<td>C2ROM1_CIDR0</td>
<td>1406</td>
</tr>
<tr>
<td>C2ROM1_CIDR1</td>
<td>1407</td>
</tr>
<tr>
<td>C2ROM1_CIDR2</td>
<td>1407</td>
</tr>
<tr>
<td>C2ROM1_CIDR3</td>
<td>1407</td>
</tr>
<tr>
<td>C2ROM1_MEMTYPEr</td>
<td>1404</td>
</tr>
<tr>
<td>C2ROM2_CIDR0</td>
<td>1412</td>
</tr>
<tr>
<td>C2ROM2_CIDR1</td>
<td>1413</td>
</tr>
<tr>
<td>C2ROM2_CIDR2</td>
<td>1413</td>
</tr>
<tr>
<td>C2ROM2_CIDR3</td>
<td>1413</td>
</tr>
<tr>
<td>C2ROM2_MEMTYPEr</td>
<td>1410</td>
</tr>
<tr>
<td>COMP1_CSR</td>
<td>521</td>
</tr>
<tr>
<td>COMP2_CSR</td>
<td>523</td>
</tr>
<tr>
<td>Index</td>
<td>RM0434</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
</tr>
<tr>
<td>CRC_CR</td>
<td>137</td>
</tr>
<tr>
<td>CRC_DR</td>
<td>136</td>
</tr>
<tr>
<td>CRC_IDR</td>
<td>136</td>
</tr>
<tr>
<td>CRC_INIT</td>
<td>138</td>
</tr>
<tr>
<td>CRC_POL</td>
<td>138</td>
</tr>
<tr>
<td>CRS_CFGR</td>
<td>143</td>
</tr>
<tr>
<td>CRS_CR</td>
<td>142</td>
</tr>
<tr>
<td>CRS_ICR</td>
<td>146</td>
</tr>
<tr>
<td>CRS_ISR</td>
<td>144</td>
</tr>
<tr>
<td>CTI_APPCLEAR</td>
<td>138</td>
</tr>
<tr>
<td>CTI_APPPULSER</td>
<td>1379</td>
</tr>
<tr>
<td>CTI_APPSETR</td>
<td>1377</td>
</tr>
<tr>
<td>CTI_AUTHSTATR</td>
<td>1384</td>
</tr>
<tr>
<td>CTI_CHINSTSR</td>
<td>1381</td>
</tr>
<tr>
<td>CTI_CHOUTSTSR</td>
<td>1381</td>
</tr>
<tr>
<td>CTI_CIDR0</td>
<td>1388</td>
</tr>
<tr>
<td>CTI_CIDR1</td>
<td>1388</td>
</tr>
<tr>
<td>CTI_CIDR2</td>
<td>1389</td>
</tr>
<tr>
<td>CTI_CIDR3</td>
<td>1389</td>
</tr>
<tr>
<td>CTI_CLAIMCLR</td>
<td>1383</td>
</tr>
<tr>
<td>CTI_CLAIMSETR</td>
<td>1382</td>
</tr>
<tr>
<td>CTI_CONTROLR</td>
<td>1377</td>
</tr>
<tr>
<td>CTI_DEVIDR</td>
<td>1385</td>
</tr>
<tr>
<td>CTI_DEVTYPER</td>
<td>1385</td>
</tr>
<tr>
<td>CTI_GATER</td>
<td>1382</td>
</tr>
<tr>
<td>CTI_INENRx</td>
<td>1379</td>
</tr>
<tr>
<td>CTI_INTACKR</td>
<td>1377</td>
</tr>
<tr>
<td>CTI_LAR</td>
<td>1383</td>
</tr>
<tr>
<td>CTI_OUTENRx</td>
<td>1380</td>
</tr>
<tr>
<td>CTI_PIDR0</td>
<td>1386</td>
</tr>
<tr>
<td>CTI_PIDR1</td>
<td>1386</td>
</tr>
<tr>
<td>CTI_PIDR2</td>
<td>1387</td>
</tr>
<tr>
<td>CTI_PIDR3</td>
<td>1387</td>
</tr>
<tr>
<td>CTI_PIDR4</td>
<td>1386</td>
</tr>
<tr>
<td>CTI_TRGISTSR</td>
<td>1380</td>
</tr>
<tr>
<td>CTI_TRGOSTSR</td>
<td>1381</td>
</tr>
<tr>
<td>DMA_ISR</td>
<td>338</td>
</tr>
<tr>
<td>DMAMUX_CFR</td>
<td>360</td>
</tr>
<tr>
<td>DMAMUX_CSR</td>
<td>360</td>
</tr>
<tr>
<td>DMAMUX_CxCR</td>
<td>359</td>
</tr>
<tr>
<td>DMAMUX_RGCIFR</td>
<td>361</td>
</tr>
<tr>
<td>DMAMUX_RGxCR</td>
<td>361</td>
</tr>
<tr>
<td>DP_ABORTR</td>
<td>1357</td>
</tr>
<tr>
<td>DP_CTRL/STATR</td>
<td>1358</td>
</tr>
<tr>
<td>DWT_CIDR0</td>
<td>1425, 1454</td>
</tr>
<tr>
<td>DWT_CIDR1</td>
<td>1425, 1454</td>
</tr>
<tr>
<td>DWT_CIDR2</td>
<td>1425, 1454</td>
</tr>
<tr>
<td>DWT_CIDR3</td>
<td>1426, 1455</td>
</tr>
<tr>
<td>DWT_COMPxR</td>
<td>1421, 1450</td>
</tr>
<tr>
<td>DWT_CPICNTR</td>
<td>1418, 1447</td>
</tr>
<tr>
<td>DWT_CTRLR</td>
<td>1416, 1445</td>
</tr>
<tr>
<td>DWT_CYCCNTR</td>
<td>1418, 1447</td>
</tr>
<tr>
<td>DWT_EXCCNTR</td>
<td>1419, 1448</td>
</tr>
<tr>
<td>DWT_FOLDCNTR</td>
<td>1420, 1449</td>
</tr>
<tr>
<td>DWT_FUNCxR</td>
<td>1421, 1450</td>
</tr>
<tr>
<td>DWT_LSUCNTR</td>
<td>1420, 1449</td>
</tr>
<tr>
<td>DWT_MaskxR</td>
<td>1421, 1450</td>
</tr>
<tr>
<td>DWT_PCSR</td>
<td>1420, 1449</td>
</tr>
<tr>
<td>DWT_PIDR0</td>
<td>1423, 1452</td>
</tr>
<tr>
<td>DWT_PIDR1</td>
<td>1423, 1452</td>
</tr>
<tr>
<td>DWT_PIDR2</td>
<td>1424, 1453</td>
</tr>
<tr>
<td>DWT_PIDR3</td>
<td>1424, 1453</td>
</tr>
<tr>
<td>DWT_PIDR4</td>
<td>1422, 1451</td>
</tr>
<tr>
<td>DWT_SLPCNTR</td>
<td>1419, 1448</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>ETM_AUTHSTATR</td>
<td>1486</td>
</tr>
<tr>
<td>ETM_CCCR</td>
<td>1480</td>
</tr>
<tr>
<td>ETM_CCR</td>
<td>1474</td>
</tr>
<tr>
<td>ETM_CIDR0</td>
<td>1489</td>
</tr>
<tr>
<td>ETM_CIDR1</td>
<td>1489</td>
</tr>
<tr>
<td>ETM_CIDR2</td>
<td>1490</td>
</tr>
<tr>
<td>ETM_CIDR3</td>
<td>1490</td>
</tr>
<tr>
<td>ETM_CLAIMCLR</td>
<td>1484</td>
</tr>
<tr>
<td>ETM_CLAIMSETR</td>
<td>1484</td>
</tr>
<tr>
<td>ETM_CTR</td>
<td>1479</td>
</tr>
<tr>
<td>ETM_CTR</td>
<td>1479</td>
</tr>
<tr>
<td>ETM_DEVTYPER</td>
<td>1486</td>
</tr>
<tr>
<td>ETM_FFLR</td>
<td>1478</td>
</tr>
<tr>
<td>ETM_IDR</td>
<td>1480</td>
</tr>
<tr>
<td>ETM_IDR</td>
<td>1480</td>
</tr>
<tr>
<td>ETM_IDR2</td>
<td>1483</td>
</tr>
<tr>
<td>ETM_LAR</td>
<td>1485</td>
</tr>
<tr>
<td>ETM_LSR</td>
<td>1485</td>
</tr>
<tr>
<td>ETM_PDSR</td>
<td>1483</td>
</tr>
<tr>
<td>ETM_PIDR0</td>
<td>1487</td>
</tr>
</tbody>
</table>

1524/1530 RM0434 Rev 13
<table>
<thead>
<tr>
<th>RM0434</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMT_TRIGGER</td>
<td>1475</td>
</tr>
<tr>
<td>EXTI_C2EMR1</td>
<td>388</td>
</tr>
<tr>
<td>EMT_TESSEICR</td>
<td>1481</td>
</tr>
<tr>
<td>EMT_TEEVR</td>
<td>1477</td>
</tr>
<tr>
<td>EMT_SYNCFR</td>
<td>1479</td>
</tr>
<tr>
<td>EMT_TRACEIDR</td>
<td>1483</td>
</tr>
<tr>
<td>EMT_TRIGGER</td>
<td>1475</td>
</tr>
<tr>
<td>EMT_TSEVR</td>
<td>1482</td>
</tr>
<tr>
<td>EXTI_C2EMR1</td>
<td>388</td>
</tr>
<tr>
<td>EXTI_C2EMR2</td>
<td>390</td>
</tr>
<tr>
<td>EXTI_C2IMR1</td>
<td>387</td>
</tr>
<tr>
<td>EXTI_C2IMR2</td>
<td>389</td>
</tr>
<tr>
<td>EXTI_EMR1</td>
<td>387</td>
</tr>
<tr>
<td>EXTI_EMR2</td>
<td>389</td>
</tr>
<tr>
<td>EXTI_FTSR1</td>
<td>382</td>
</tr>
<tr>
<td>EXTI_FTSR2</td>
<td>384</td>
</tr>
<tr>
<td>EXTI_IMR1</td>
<td>386</td>
</tr>
<tr>
<td>EXTI_IMR2</td>
<td>388</td>
</tr>
<tr>
<td>EXTI_PR1</td>
<td>383</td>
</tr>
<tr>
<td>EXTI_PR2</td>
<td>386</td>
</tr>
<tr>
<td>EXTI_RTSR1</td>
<td>381</td>
</tr>
<tr>
<td>EXTI_RTSR2</td>
<td>383</td>
</tr>
<tr>
<td>EXTI_SWIER1</td>
<td>382</td>
</tr>
<tr>
<td>EXTI_SWIER2</td>
<td>385</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_ACR</td>
<td>111</td>
</tr>
<tr>
<td>FLASH_C2ACR</td>
<td>122</td>
</tr>
<tr>
<td>FLASH_C2CR</td>
<td>125</td>
</tr>
<tr>
<td>FLASH_C2SR</td>
<td>123</td>
</tr>
<tr>
<td>FLASH_CR</td>
<td>114</td>
</tr>
<tr>
<td>FLASH_ECCR</td>
<td>116</td>
</tr>
<tr>
<td>FLASH_IPCCBR</td>
<td>122</td>
</tr>
<tr>
<td>FLASH_KEYR</td>
<td>112</td>
</tr>
<tr>
<td>FLASH_OPTKEYR</td>
<td>112</td>
</tr>
<tr>
<td>FLASH_OTPR</td>
<td>117</td>
</tr>
<tr>
<td>FLASH_PCROP1AER</td>
<td>120</td>
</tr>
<tr>
<td>FLASH_PCROP1ASR</td>
<td>119</td>
</tr>
<tr>
<td>FLASH_PCROP1BER</td>
<td>122</td>
</tr>
<tr>
<td>FLASH_PCROP1BSR</td>
<td>121</td>
</tr>
<tr>
<td>FLASH_SFR</td>
<td>126</td>
</tr>
<tr>
<td>FLASH_SR</td>
<td>113</td>
</tr>
<tr>
<td>FLASH_SRRVR</td>
<td>127</td>
</tr>
<tr>
<td>FLASH_WRP1AR</td>
<td>120</td>
</tr>
<tr>
<td>FLASH_WRP1BR</td>
<td>121</td>
</tr>
<tr>
<td>FPB_CIDR0</td>
<td>1470</td>
</tr>
<tr>
<td>FPB_CIDR1</td>
<td>1470</td>
</tr>
<tr>
<td>FPB_CIDR2</td>
<td>1470</td>
</tr>
<tr>
<td>FPB_CIDR3</td>
<td>1471</td>
</tr>
<tr>
<td>FPB_COMPxR</td>
<td>1467</td>
</tr>
<tr>
<td>FPB_CTRLR</td>
<td>1466</td>
</tr>
<tr>
<td>FPB_PIDR0</td>
<td>1468</td>
</tr>
<tr>
<td>FPB_PIDR1</td>
<td>1468</td>
</tr>
<tr>
<td>FPB_PIDR2</td>
<td>1469</td>
</tr>
<tr>
<td>FPB_PIDR3</td>
<td>1469</td>
</tr>
<tr>
<td>FPB_PIDR4</td>
<td>1467</td>
</tr>
<tr>
<td>FPB_REMAPR</td>
<td>1466</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>G</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIOx_AFRH</td>
<td>306</td>
</tr>
<tr>
<td>GPIOx_AFRL</td>
<td>305</td>
</tr>
<tr>
<td>GPIOx_BRR</td>
<td>307</td>
</tr>
<tr>
<td>GPIOx_BSRR</td>
<td>303</td>
</tr>
<tr>
<td>GPIOx_IDR</td>
<td>302</td>
</tr>
<tr>
<td>GPIOx_LCKR</td>
<td>304</td>
</tr>
<tr>
<td>GPIOx_MODER</td>
<td>300</td>
</tr>
<tr>
<td>GPIOx_ODR</td>
<td>303</td>
</tr>
<tr>
<td>GPIOx_OOSPEEDR</td>
<td>301</td>
</tr>
<tr>
<td>GPIOx_OTYPER</td>
<td>301</td>
</tr>
<tr>
<td>GPIOx_PUPDR</td>
<td>302</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HSEM_CnICR</td>
<td>1299</td>
</tr>
<tr>
<td>HSEM_CnIER</td>
<td>1299</td>
</tr>
<tr>
<td>HSEM_CnISR</td>
<td>1299</td>
</tr>
<tr>
<td>HSEM_CnMISR</td>
<td>1300</td>
</tr>
<tr>
<td>HSEM_CR</td>
<td>1300</td>
</tr>
<tr>
<td>HSEM_KEYR</td>
<td>1301</td>
</tr>
<tr>
<td>HSEM_RLRx</td>
<td>1298</td>
</tr>
<tr>
<td>HSEM_Rx</td>
<td>1297</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_CR1</td>
<td>1021</td>
</tr>
<tr>
<td>I2C_CR2</td>
<td>1023</td>
</tr>
<tr>
<td>I2C_ICR</td>
<td>1031</td>
</tr>
<tr>
<td>I2C_ISR</td>
<td>1029</td>
</tr>
<tr>
<td>I2C_OAR1</td>
<td>1025</td>
</tr>
<tr>
<td>I2C_OAR2</td>
<td>1026</td>
</tr>
<tr>
<td>I2C_PECR</td>
<td>1032</td>
</tr>
<tr>
<td>I2C_RXDR</td>
<td>1032</td>
</tr>
<tr>
<td>I2C_TIMEOUTR</td>
<td>1028</td>
</tr>
<tr>
<td>I2C_TIMINGR</td>
<td>1027</td>
</tr>
<tr>
<td>I2C_TXDR</td>
<td>1033</td>
</tr>
<tr>
<td>IPCC_C1CR</td>
<td>1284</td>
</tr>
<tr>
<td>IPCC_C1MR</td>
<td>1284</td>
</tr>
<tr>
<td>IPCC_C1SCR</td>
<td>1285</td>
</tr>
<tr>
<td>Name</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------</td>
<td>------</td>
</tr>
<tr>
<td>IPCC_C1TOC2SR</td>
<td>1286</td>
</tr>
<tr>
<td>IPCC_C2CR</td>
<td>1286</td>
</tr>
<tr>
<td>IPCC_C2MR</td>
<td>1287</td>
</tr>
<tr>
<td>IPCC_C2SCR</td>
<td>1287</td>
</tr>
<tr>
<td>IPCC_C2TOC1SR</td>
<td>1288</td>
</tr>
<tr>
<td>ITM_CIDR0</td>
<td>1463</td>
</tr>
<tr>
<td>ITM_CIDR1</td>
<td>1463</td>
</tr>
<tr>
<td>ITM_CIDR2</td>
<td>1463</td>
</tr>
<tr>
<td>ITM_CIDR3</td>
<td>1464</td>
</tr>
<tr>
<td>ITM_PIDR0</td>
<td>1461</td>
</tr>
<tr>
<td>ITM_PIDR1</td>
<td>1461</td>
</tr>
<tr>
<td>ITM_PIDR2</td>
<td>1462</td>
</tr>
<tr>
<td>ITM_PIDR3</td>
<td>1462</td>
</tr>
<tr>
<td>ITM_PIDR4</td>
<td>1460</td>
</tr>
<tr>
<td>ITM_STIMRx</td>
<td>1458</td>
</tr>
<tr>
<td>ITM_TCR</td>
<td>1459</td>
</tr>
<tr>
<td>ITM_TCRx</td>
<td>1458</td>
</tr>
<tr>
<td>ITM_TPR</td>
<td>1459</td>
</tr>
<tr>
<td>IWDG_KR</td>
<td>960</td>
</tr>
<tr>
<td>IWDG_PR</td>
<td>961</td>
</tr>
<tr>
<td>IWDG_RLR</td>
<td>962</td>
</tr>
<tr>
<td>IWDG_SR</td>
<td>963</td>
</tr>
<tr>
<td>IWDG_WINR</td>
<td>964</td>
</tr>
<tr>
<td>LCD_CLR</td>
<td>554</td>
</tr>
<tr>
<td>LCD_CR</td>
<td>549</td>
</tr>
<tr>
<td>LCD_FCR</td>
<td>550</td>
</tr>
<tr>
<td>LCD_RAMx</td>
<td>554-555</td>
</tr>
<tr>
<td>LCD_SR</td>
<td>552</td>
</tr>
<tr>
<td>LPTIM_ARR</td>
<td>910</td>
</tr>
<tr>
<td>LPTIM_CFGR</td>
<td>905</td>
</tr>
<tr>
<td>LPTIM_CMP</td>
<td>910</td>
</tr>
<tr>
<td>LPTIM_CNT</td>
<td>911</td>
</tr>
<tr>
<td>LPTIM_CR</td>
<td>908</td>
</tr>
<tr>
<td>LPTIM_ICR</td>
<td>904</td>
</tr>
<tr>
<td>LPTIM_IER</td>
<td>904</td>
</tr>
<tr>
<td>LPTIM_ISR</td>
<td>903</td>
</tr>
<tr>
<td>LPTIM1_OR</td>
<td>911</td>
</tr>
<tr>
<td>LPTIM2_OR</td>
<td>912</td>
</tr>
<tr>
<td>LPUART_BRR</td>
<td>1163</td>
</tr>
<tr>
<td>LPUART_CR1</td>
<td>1152, 1155</td>
</tr>
<tr>
<td>LPUART_CR2</td>
<td>1158</td>
</tr>
<tr>
<td>LPUART_CR3</td>
<td>1160</td>
</tr>
<tr>
<td>LPUART_ICR</td>
<td>1172</td>
</tr>
<tr>
<td>LPUART_ISR</td>
<td>1164, 1169</td>
</tr>
<tr>
<td>LPUART_PRESC</td>
<td>1174</td>
</tr>
<tr>
<td>LPUART_RDR</td>
<td>1173</td>
</tr>
<tr>
<td>LPUART_RQR</td>
<td>1164</td>
</tr>
<tr>
<td>LPUART_TDR</td>
<td>1173</td>
</tr>
<tr>
<td>PKA_CLRFR</td>
<td>662</td>
</tr>
<tr>
<td>PKA_CR</td>
<td>660</td>
</tr>
<tr>
<td>PKA_SR</td>
<td>661</td>
</tr>
<tr>
<td>PWR_C2CR1</td>
<td>189</td>
</tr>
<tr>
<td>PWR_C2CR3</td>
<td>191</td>
</tr>
<tr>
<td>PWR_CR1</td>
<td>174</td>
</tr>
<tr>
<td>PWR_CR2</td>
<td>175</td>
</tr>
<tr>
<td>PWR_CR3</td>
<td>176</td>
</tr>
<tr>
<td>PWR_CR4</td>
<td>178</td>
</tr>
<tr>
<td>PWR_CR5</td>
<td>182</td>
</tr>
<tr>
<td>PWR_EXTSCR</td>
<td>192</td>
</tr>
<tr>
<td>PWR_PDCRA</td>
<td>184</td>
</tr>
<tr>
<td>PWR_PDCRB</td>
<td>185</td>
</tr>
<tr>
<td>PWR_PDCRC</td>
<td>186</td>
</tr>
<tr>
<td>PWR_PDCRD</td>
<td>187</td>
</tr>
<tr>
<td>PWR_PDCRE</td>
<td>188</td>
</tr>
<tr>
<td>PWR_PDCRH</td>
<td>189</td>
</tr>
<tr>
<td>PWR_PUCRA</td>
<td>183</td>
</tr>
<tr>
<td>PWR_PUCRB</td>
<td>184</td>
</tr>
<tr>
<td>PWR_PUCRC</td>
<td>185</td>
</tr>
<tr>
<td>PWR_PUCRD</td>
<td>186</td>
</tr>
<tr>
<td>PWR_PUCRE</td>
<td>187</td>
</tr>
<tr>
<td>PWR_PUCRH</td>
<td>188</td>
</tr>
<tr>
<td>PWR_SCR</td>
<td>181</td>
</tr>
<tr>
<td>PWR_SR1</td>
<td>179</td>
</tr>
<tr>
<td>PWR_SR2</td>
<td>180</td>
</tr>
<tr>
<td>QUADSPI_ABR</td>
<td>414</td>
</tr>
<tr>
<td>QUADSPI_AR</td>
<td>414</td>
</tr>
<tr>
<td>QUADSPI_CCR</td>
<td>412</td>
</tr>
<tr>
<td>QUADSPI_CR</td>
<td>407</td>
</tr>
<tr>
<td>QUADSPI_DCR</td>
<td>409</td>
</tr>
<tr>
<td>QUADSPI_DLR</td>
<td>412</td>
</tr>
<tr>
<td>QUADSPI_DR</td>
<td>415</td>
</tr>
<tr>
<td>QUADSPI_FCR</td>
<td>411</td>
</tr>
<tr>
<td>QUADSPI_LPTR</td>
<td>417</td>
</tr>
<tr>
<td>QUADSPI_PIR</td>
<td>416</td>
</tr>
<tr>
<td>QUADSPI_PSMAR</td>
<td>416</td>
</tr>
<tr>
<td>QUADSPI_PSMIKR</td>
<td>415</td>
</tr>
<tr>
<td>QUADSPI_SR</td>
<td>410</td>
</tr>
<tr>
<td>RCC_AHB1ENR</td>
<td>247, 270</td>
</tr>
<tr>
<td>RCC_AHB1RSTR</td>
<td>239-240</td>
</tr>
<tr>
<td>RCC_AHB1SMENR</td>
<td>253, 277</td>
</tr>
<tr>
<td>RCC_AHB2ENR</td>
<td>248, 271</td>
</tr>
<tr>
<td>RCC_AHB2RSTR</td>
<td>241</td>
</tr>
<tr>
<td>RCC_AHB2SMENR</td>
<td>254, 278</td>
</tr>
<tr>
<td>RCC_AHB3ENR</td>
<td>249, 272</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>RCC_AHB3RSTR</td>
<td>242</td>
</tr>
<tr>
<td>RCC_AHB3SMENR</td>
<td>256, 280</td>
</tr>
<tr>
<td>RCC_APB1ENR1</td>
<td>250, 273</td>
</tr>
<tr>
<td>RCC_APB1ENR2</td>
<td>252, 275-276, 284</td>
</tr>
<tr>
<td>RCC_APB1RSTR1</td>
<td>243</td>
</tr>
<tr>
<td>RCC_APB1RSTR2</td>
<td>245</td>
</tr>
<tr>
<td>RCC_APB1SMENR1</td>
<td>257, 281</td>
</tr>
<tr>
<td>RCC_APB1SMENR2</td>
<td>259, 282</td>
</tr>
<tr>
<td>RCC_APB2ENR</td>
<td>252, 275</td>
</tr>
<tr>
<td>RCC_APB2RSTR</td>
<td>245</td>
</tr>
<tr>
<td>RCC_APB2SMENR</td>
<td>259, 283</td>
</tr>
<tr>
<td>RCC_BDCR</td>
<td>262</td>
</tr>
<tr>
<td>RCC_CICR</td>
<td>267</td>
</tr>
<tr>
<td>RCC_CIFR</td>
<td>268</td>
</tr>
<tr>
<td>RCC_CSR</td>
<td>264</td>
</tr>
<tr>
<td>RCC_CR</td>
<td>223</td>
</tr>
<tr>
<td>RCC_CRRCR</td>
<td>266-268</td>
</tr>
<tr>
<td>RCC_CSR</td>
<td>264</td>
</tr>
<tr>
<td>RCC_ICSCR</td>
<td>226</td>
</tr>
<tr>
<td>RCC_PLLCFGR</td>
<td>229</td>
</tr>
<tr>
<td>RCC_PLLSAI1CFGR</td>
<td>233</td>
</tr>
<tr>
<td>RNG_CR</td>
<td>586</td>
</tr>
<tr>
<td>RNG_DR</td>
<td>587</td>
</tr>
<tr>
<td>RNG_SR</td>
<td>586</td>
</tr>
<tr>
<td>RTC_ALRMAR</td>
<td>941</td>
</tr>
<tr>
<td>RTC_ALRMASSR</td>
<td>952</td>
</tr>
<tr>
<td>RTC_ALRMBSSR</td>
<td>953</td>
</tr>
<tr>
<td>RTC_BKPDR</td>
<td>954</td>
</tr>
<tr>
<td>RTC_CALR</td>
<td>948</td>
</tr>
<tr>
<td>RTC_CR</td>
<td>933</td>
</tr>
<tr>
<td>RTC_DRR</td>
<td>932</td>
</tr>
<tr>
<td>RTC_ISR</td>
<td>936</td>
</tr>
<tr>
<td>RTC_OR</td>
<td>954</td>
</tr>
<tr>
<td>RTC_PRER</td>
<td>939</td>
</tr>
<tr>
<td>RTC_SHIFTR</td>
<td>944</td>
</tr>
<tr>
<td>RTC_SSR</td>
<td>943</td>
</tr>
<tr>
<td>RTC_TAMPCTRL</td>
<td>949</td>
</tr>
<tr>
<td>RTC_TR</td>
<td>931</td>
</tr>
<tr>
<td>RTC_TSDR</td>
<td>946</td>
</tr>
<tr>
<td>RTC_TSSSR</td>
<td>947</td>
</tr>
<tr>
<td>RTC_TSTTR</td>
<td>945</td>
</tr>
<tr>
<td>RTC_WPR</td>
<td>943</td>
</tr>
<tr>
<td>RTC_WUTR</td>
<td>940</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SAI_ACLRFR</td>
<td>1258</td>
</tr>
<tr>
<td>SAI_ACR1</td>
<td>1248</td>
</tr>
<tr>
<td>INDEX</td>
<td>PAGE</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>TIM1_CCR3</td>
<td>748</td>
</tr>
<tr>
<td>TIM1_CCR4</td>
<td>749</td>
</tr>
<tr>
<td>TIM1_CCR5</td>
<td>756</td>
</tr>
<tr>
<td>TIM1_CCR6</td>
<td>757</td>
</tr>
<tr>
<td>TIM1_CNT</td>
<td>746</td>
</tr>
<tr>
<td>TIM1_CCR1</td>
<td>725</td>
</tr>
<tr>
<td>TIM1_CCR2</td>
<td>726</td>
</tr>
<tr>
<td>TIM1_DCR</td>
<td>753</td>
</tr>
<tr>
<td>TIM1_DIER</td>
<td>731</td>
</tr>
<tr>
<td>TIM1_DMAR</td>
<td>754</td>
</tr>
<tr>
<td>TIM1_EGR</td>
<td>735</td>
</tr>
<tr>
<td>TIM1_OR1</td>
<td>757</td>
</tr>
<tr>
<td>TIM1_PSC</td>
<td>746</td>
</tr>
<tr>
<td>TIM1_RCR</td>
<td>747</td>
</tr>
<tr>
<td>TIM1_SMCR</td>
<td>729</td>
</tr>
<tr>
<td>TIM1_SR</td>
<td>733</td>
</tr>
<tr>
<td>TIM1_TISEL</td>
<td>761</td>
</tr>
<tr>
<td>TIM17_AF1</td>
<td>883</td>
</tr>
<tr>
<td>TIM17_CR1</td>
<td>860</td>
</tr>
<tr>
<td>TIM17_AF1</td>
<td>883</td>
</tr>
<tr>
<td>TIM17_TISEL</td>
<td>887</td>
</tr>
<tr>
<td>TIM2_AF1</td>
<td>832</td>
</tr>
<tr>
<td>TIM2_ARR</td>
<td>829</td>
</tr>
<tr>
<td>TIM2_CCMR2</td>
<td>824-825</td>
</tr>
<tr>
<td>TIM2_CCR1</td>
<td>829</td>
</tr>
<tr>
<td>TIM2_CCR2</td>
<td>829</td>
</tr>
<tr>
<td>TIM2_CCR3</td>
<td>830</td>
</tr>
<tr>
<td>TIM2_CCR4</td>
<td>830</td>
</tr>
<tr>
<td>TIM2_CNT</td>
<td>827-828</td>
</tr>
<tr>
<td>TIM2_CCR2</td>
<td>829</td>
</tr>
<tr>
<td>TIM2_CCR2</td>
<td>829</td>
</tr>
<tr>
<td>TIM2_CCR3</td>
<td>830</td>
</tr>
<tr>
<td>TIM2_CCR4</td>
<td>830</td>
</tr>
<tr>
<td>TIM2_CNT</td>
<td>827-828</td>
</tr>
<tr>
<td>TIM2_CCR1</td>
<td>829</td>
</tr>
<tr>
<td>TIM2_CCR2</td>
<td>829</td>
</tr>
<tr>
<td>TIM2_DCR</td>
<td>831</td>
</tr>
<tr>
<td>TIM2_DIER</td>
<td>816</td>
</tr>
<tr>
<td>TIM2_DMAR</td>
<td>832</td>
</tr>
<tr>
<td>TIM2_EGR</td>
<td>819</td>
</tr>
<tr>
<td>TIM2_OR1</td>
<td>832</td>
</tr>
<tr>
<td>TIM2_PSC</td>
<td>828</td>
</tr>
<tr>
<td>TIM2_SMCR</td>
<td>813</td>
</tr>
<tr>
<td>TIM2_SR</td>
<td>817</td>
</tr>
<tr>
<td>TIM2_TISEL</td>
<td>833</td>
</tr>
<tr>
<td>TIMx_ARR</td>
<td>877</td>
</tr>
<tr>
<td>TIMx_BDTR</td>
<td>879</td>
</tr>
<tr>
<td>TIMx_CCR1</td>
<td>874</td>
</tr>
<tr>
<td>TIMx_CCMR1</td>
<td>.871-872</td>
</tr>
<tr>
<td>TIMx_CNT</td>
<td>876</td>
</tr>
<tr>
<td>TIMx_CR1</td>
<td>866</td>
</tr>
<tr>
<td>TIMx_CR2</td>
<td>867</td>
</tr>
</tbody>
</table>

**U**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART_BRR</td>
<td>1103</td>
</tr>
<tr>
<td>USART_CR1</td>
<td>1087, 1091</td>
</tr>
<tr>
<td>USART_CR2</td>
<td>1094</td>
</tr>
<tr>
<td>USART_CR3</td>
<td>1098</td>
</tr>
<tr>
<td>USART_GTPR</td>
<td>1103</td>
</tr>
<tr>
<td>USART_ICR</td>
<td>1117</td>
</tr>
<tr>
<td>USART_ISR</td>
<td>1106, 1112</td>
</tr>
<tr>
<td>USART_PRES</td>
<td>1120</td>
</tr>
<tr>
<td>USART_RDR</td>
<td>1119</td>
</tr>
<tr>
<td>USART_RQR</td>
<td>1105</td>
</tr>
<tr>
<td>USART_RTOR</td>
<td>1104</td>
</tr>
<tr>
<td>USART_TDR</td>
<td>1119</td>
</tr>
<tr>
<td>Symbol</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
</tr>
<tr>
<td>USB_ADDRn_RX</td>
<td>1333</td>
</tr>
<tr>
<td>USB_ADDRn_TX</td>
<td>1332</td>
</tr>
<tr>
<td>USB_BCDR</td>
<td>1326</td>
</tr>
<tr>
<td>USB_BTABLE</td>
<td>1325</td>
</tr>
<tr>
<td>USB_CNTR</td>
<td>1319</td>
</tr>
<tr>
<td>USB_COUNTn_RX</td>
<td>1333</td>
</tr>
<tr>
<td>USB_COUNTn_TX</td>
<td>1332</td>
</tr>
<tr>
<td>USB_DADDR</td>
<td>1324</td>
</tr>
<tr>
<td>USB_EPnR</td>
<td>1327</td>
</tr>
<tr>
<td>USB_FNR</td>
<td>1324</td>
</tr>
<tr>
<td>USBISTR</td>
<td>1321</td>
</tr>
<tr>
<td>USB_LPMCSR</td>
<td>1325</td>
</tr>
<tr>
<td>VREFBUF_CCR</td>
<td>512</td>
</tr>
<tr>
<td>VREFBUF_CSR</td>
<td>511</td>
</tr>
<tr>
<td>WWDG_CFR</td>
<td>970</td>
</tr>
<tr>
<td>WWDG_CR</td>
<td>969</td>
</tr>
<tr>
<td>WWDG_SR</td>
<td>971</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2024 STMicroelectronics – All rights reserved