RM0041
Reference manual

STM32F100xx
advanced ARM-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F100x4, STM32F100x6, STM32F100x8, STM32F100xB, STM32F100xC, STM32F100xD and STM32F100xE microcontroller memory and peripherals. The STM32F100x4, STM32F100x6, STM32F100x8, STM32F100xB, STM32F100xC, STM32F100xD and STM32F100xE will be referred to as STM32F100xx throughout the document, unless otherwise specified.

The STM32F100xx is a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics please refer to the *STM32F100xx* datasheet.

For information on programming, erasing and protection of the internal Flash memory please refer to the *STM32F100xx Flash programming manual (PM0063).*

For information on the ARM Cortex™-M3 core, please refer to the *Cortex™-M3 Technical Reference Manual*.

Related documents

Available from *www.arm.com*:

- *Cortex™-M3 Technical Reference Manual, available from:*
http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337e/DDI0337E_cortex_m3_r1p1_trm.pdf

Available from *your STMicroelectronics sales office*:

- *STM32F100xx datasheet*
- *STM32F100xx Flash programming manual*

Contents

1 **Documentation conventions 32**

1.1 List of abbreviations for registers 32

1.2 Glossary 32

1.3 Peripheral availability 32

2 **Memory and bus architecture 33**

2.1 System architecture 33

2.2 Memory organization 35

2.3 Memory map 36

2.3.1 Embedded SRAM 39

2.3.2 Bit banding 40

2.3.3 Embedded Flash memory 40

2.4 Boot configuration 44

3 **CRC calculation unit 46**

3.1 CRC introduction 46

3.2 CRC main features 46

3.3 CRC functional description 47

3.4 CRC registers 47

3.4.1 Data register (CRC_DR) 47

3.4.2 Independent data register (CRC_IDR) 47

3.4.3 Control register (CRC_CR) 48

3.4.4 CRC register map 48

4 **Power control (PWR) 49**

4.1 Power supplies 49

4.1.1 Independent A/D and D/A converter supply and reference voltage 50

4.1.2 Battery backup domain 50

4.1.3 Voltage regulator 51

4.2 Power supply supervisor 51

4.2.1 Power on reset (POR)/power down reset (PDR) 51

4.2.2 Programmable voltage detector (PVD) 52

4.3 Low-power modes 53

4.3.1 Slowing down system clocks . . . 53
4.3.2 Peripheral clock gating . . . 54
4.3.3 Sleep mode . . . 54
4.3.4 Stop mode . . . 55
4.3.5 Standby mode . . . 56
4.3.6 Auto-wakeup (AWU) from low-power mode . . . 58
4.4 Power control registers . . . 58
4.4.1 Power control register (PWR_CR) . . . 58
4.4.2 Power control/status register (PWR_CSR) . . . 60
4.4.3 PWR register map . . . 61

5 Backup registers (BKP) . . . 62
5.1 BKP introduction . . . 62
5.2 BKP main features . . . 62
5.3 BKP functional description . . . 63
5.3.1 Tamper detection . . . 63
5.3.2 RTC calibration . . . 63
5.4 BKP registers . . . 64
5.4.1 Backup data register x (BKP_DRx) (x = 1 ..42) . . . 64
5.4.2 RTC clock calibration register (BKP_RTCCR) . . . 64
5.4.3 Backup control register (BKP_CR) . . . 65
5.4.4 Backup control/status register (BKP_CSR) . . . 65
5.4.5 BKP register map . . . 66

6 Reset and clock control (RCC) . . . 68
6.1 Reset . . . 68
6.1.1 System reset . . . 68
6.1.2 Power reset . . . 69
6.1.3 Backup domain reset . . . 69
6.2 Clocks . . . 69
6.2.1 HSE clock . . . 72
6.2.2 HSI clock . . . 73
6.2.3 PLL . . . 73
6.2.4 LSE clock . . . 74
6.2.5 LSI clock . . . 74
6.2.6 System clock (SYSCLK) selection . . . 75

6.2.7 Clock security system (CSS) . . . 75
6.2.8 RTC clock . . . 75
6.2.9 Watchdog clock . . . 76
6.2.10 Clock-out capability . . . 76
6.3 RCC registers . . . 77
6.3.1 Clock control register (RCC_CR) . . . 77
6.3.2 Clock configuration register (RCC_CFGR) . . . 79
6.3.3 Clock interrupt register (RCC_CIR) . . . 81
6.3.4 APB2 peripheral reset register (RCC_APB2RSTR) . . . 83
6.3.5 APB1 peripheral reset register (RCC_APB1RSTR) . . . 85
6.3.6 AHB peripheral clock enable register (RCC_AHBENR) . . . 87
6.3.7 APB2 peripheral clock enable register (RCC_APB2ENR) . . . 89
6.3.8 APB1 peripheral clock enable register (RCC_APB1ENR) . . . 91
6.3.9 Backup domain control register (RCC_BDCR) . . . 94
6.3.10 Control/status register (RCC_CSR) . . . 95
6.3.11 Clock configuration register2 (RCC_CFGR2) . . . 97
6.3.12 RCC register map . . . 98

7 General-purpose and alternate-function I/Os (GPIOs and AFIOs) . . . 99
7.1 GPIO functional description . . . 99
7.1.1 General-purpose I/O (GPIO) . . . 101
7.1.2 Atomic bit set or reset . . . 101
7.1.3 External interrupt/wakeup lines . . . 102
7.1.4 Alternate functions (AF) . . . 102
7.1.5 Software remapping of I/O alternate functions . . . 102
7.1.6 GPIO locking mechanism . . . 102
7.1.7 Input configuration . . . 103
7.1.8 Output configuration . . . 103
7.1.9 Alternate function configuration . . . 104
7.1.10 Analog configuration . . . 105
7.1.11 GPIO configurations for device peripherals . . . 106
7.2 GPIO registers . . . 110
7.2.1 Port configuration register low (GPIOx_CRL) (x=A..G) . . . 110
7.2.2 Port configuration register high (GPIOx_CRH) (x=A..G) . . . 111
7.2.3 Port input data register (GPIOx_IDR) (x=A..G) . . . 111
7.2.4 Port output data register (GPIOx_ODR) (x=A..G) . . . 112
7.2.5 Port bit set/reset register (GPIOx_BSRR) (x=A..G) . . . 112

7.2.6 Port bit reset register (GPIOx_BRR) (x=A..G) 113
7.2.7 Port configuration lock register (GPIOx_LCKR) (x=A..G) 113
7.3 Alternate function I/O and debug configuration (AFIO) 114
7.3.1 Using OSC32_IN/OSC32_OUT pins as GPIO ports PC14/PC15 114
7.3.2 Using OSC_IN/OSC_OUT pins as GPIO ports PD0/PD1 114
7.3.3 JTAG/SWD alternate function remapping 115
7.3.4 Timer alternate function remapping 115
7.3.5 USART alternate function remapping 118
7.3.6 I2C1 alternate function remapping 119
7.3.7 SPI1 alternate function remapping 119
7.3.8 CEC remap 119
7.4 AFIO registers 119
7.4.1 Event control register (AFIO_EVCR) 120
7.4.2 AF remap and debug I/O configuration register (AFIO_MAPR) 121
7.4.3 External interrupt configuration register 1 (AFIO_EXTICR1) 123
7.4.4 External interrupt configuration register 2 (AFIO_EXTICR2) 123
7.4.5 External interrupt configuration register 3 (AFIO_EXTICR3) 124
7.4.6 External interrupt configuration register 4 (AFIO_EXTICR4) 124
7.4.7 AF remap and debug I/O configuration register (AFIO_MAPR2) 125
7.5 GPIO and AFIO register maps 127

8 Interrupts and events 129

8.1 Nested vectored interrupt controller (NVIC) 129
8.1.1 SysTick calibration value register 129
8.1.2 Interrupt and exception vectors 129
8.2 External interrupt/event controller (EXTI) 132
8.2.1 Main features 132
8.2.2 Block diagram 132
8.2.3 Wakeup event management 133
8.2.4 Functional description 133
8.2.5 External interrupt/event line mapping 134
8.3 EXTI registers 136
8.3.1 Interrupt mask register (EXTI_IMR) 136
8.3.2 Event mask register (EXTI_EMR) 136
8.3.3 Rising trigger selection register (EXTI_RTSR) 137
8.3.4 Falling trigger selection register (EXTI_FTSR) 137

8.3.5 Software interrupt event register (EXTI_SWIER) . . . 138
8.3.6 Pending register (EXTI_PR) . . . 138
8.3.7 EXTI register map . . . 139

9 Direct memory access controller (DMA) . . . 140
9.1 DMA introduction . . . 140
9.2 DMA main features . . . 140
9.3 DMA functional description . . . 142
9.3.1 DMA transactions . . . 142
9.3.2 Arbiter . . . 143
9.3.3 DMA channels . . . 143
9.3.4 Programmable data width, data alignment and endians . . . 145
9.3.5 Error management . . . 146
9.3.6 Interrupts . . . 146
9.3.7 DMA request mapping . . . 146
9.4 DMA registers . . . 151
9.4.1 DMA interrupt status register (DMA_ISR) . . . 151
9.4.2 DMA interrupt flag clear register (DMA_IFCR) . . . 152
9.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1..7, where x = channel number) . . . 153
9.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1..7), where x = channel number) . . . 154
9.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1..7), where x = channel number) . . . 155
9.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1..7), where x = channel number) . . . 155
9.4.7 DMA register map . . . 156

10 Analog-to-digital converter (ADC) . . . 158
10.1 ADC introduction . . . 158
10.2 ADC main features . . . 158
10.3 ADC functional description . . . 159
10.3.1 ADC on-off control . . . 161
10.3.2 ADC clock . . . 161
10.3.3 Channel selection . . . 161
10.3.4 Single conversion mode . . . 161
10.3.5 Continuous conversion mode . . . 162
10.3.6 Timing diagram . . . 162

10.3.7 Analog watchdog . . . 163
10.3.8 Scan mode . . . 163
10.3.9 Injected channel management . . . 164
10.3.10 Discontinuous mode . . . 165
10.4 Calibration . . . 165
10.5 Data alignment . . . 166
10.6 Channel-by-channel programmable sample time . . . 167
10.7 Conversion on external trigger . . . 167
10.8 DMA request . . . 168
10.9 Temperature sensor . . . 168
10.10 ADC interrupts . . . 170
10.11 ADC registers . . . 171
10.11.1 ADC status register (ADC_SR) . . . 171
10.11.2 ADC control register 1 (ADC_CR1) . . . 172
10.11.3 ADC control register 2 (ADC_CR2) . . . 173
10.11.4 ADC sample time register 1 (ADC_SMPR1) . . . 176
10.11.5 ADC sample time register 2 (ADC_SMPR2) . . . 177
10.11.6 ADC injected channel data offset register x (ADC_JOFRx)(x=1..4) . . 177
10.11.7 ADC watchdog high threshold register (ADC_HTR) . . . 178
10.11.8 ADC watchdog low threshold register (ADC_LTR) . . . 178
10.11.9 ADC regular sequence register 1 (ADC_SQR1) . . . 179
10.11.10 ADC regular sequence register 2 (ADC_SQR2) . . . 180
10.11.11 ADC regular sequence register 3 (ADC_SQR3) . . . 181
10.11.12 ADC injected sequence register (ADC_JSQR) . . . 182
10.11.13 ADC injected data register x (ADC_JDRx) (x= 1..4) . . . 183
10.11.14 ADC regular data register (ADC_DR) . . . 183
10.11.15 ADC register map . . . 184

11 Digital-to-analog converter (DAC) . . . 186
11.1 DAC introduction . . . 186
11.2 DAC main features . . . 186
11.3 DAC functional description . . . 188
11.3.1 DAC channel enable . . . 188
11.3.2 DAC output buffer enable . . . 188
11.3.3 DAC data format . . . 188
11.3.4 DAC conversion . . . 189

11.3.5 DAC output voltage . . . 190
11.3.6 DAC trigger selection . . . 190
11.3.7 DMA request . . . 190
11.3.8 Noise generation . . . 191
11.3.9 Triangle-wave generation . . . 192
11.4 Dual DAC channel conversion . . . 193
11.4.1 Independent trigger without wave generation . . . 193
11.4.2 Independent trigger with single LFSR generation . . . 194
11.4.3 Independent trigger with different LFSR generation . . . 194
11.4.4 Independent trigger with single triangle generation . . . 194
11.4.5 Independent trigger with different triangle generation . . . 195
11.4.6 Simultaneous software start . . . 195
11.4.7 Simultaneous trigger without wave generation . . . 195
11.4.8 Simultaneous trigger with single LFSR generation . . . 196
11.4.9 Simultaneous trigger with different LFSR generation . . . 196
11.4.10 Simultaneous trigger with single triangle generation . . . 196
11.4.11 Simultaneous trigger with different triangle generation . . . 197
11.5 DAC registers . . . 197
11.5.1 DAC control register (DAC_CR) . . . 197
11.5.2 DAC software trigger register (DAC_SWTRIGR) . . . 200
11.5.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) . . . 201
11.5.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) . . . 201
11.5.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) . . . 201
11.5.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) . . . 202
11.5.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) . . . 202
11.5.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) . . . 202
11.5.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) . . . 203
11.5.10 DUAL DAC 12-bit left aligned data holding register (DAC_DHR12LD) . . . 203
11.5.11 DUAL DAC 8-bit right aligned data holding register (DAC_DHR8RD) . . . 204
11.5.12 DAC channel1 data output register (DAC_DOR1) . . . 204

11.5.13 DAC channel2 data output register (DAC_DOR2) 204
11.5.14 DAC status register (DAC_SR) 205
11.5.15 DAC register map 206

12 Advanced-control timers (TIM1) 207
12.1 TIM1 introduction 207
12.2 TIM1 main features 207
12.3 TIM1 functional description 210
12.3.1 Time-base unit 210
12.3.2 Counter modes 211
12.3.3 Repetition counter 219
12.3.4 Clock selection 221
12.3.5 Capture/compare channels 223
12.3.6 Input capture mode 225
12.3.7 PWM input mode 226
12.3.8 Forced output mode 227
12.3.9 Output compare mode 228
12.3.10 PWM mode 229
12.3.11 Complementary outputs and dead-time insertion 232
12.3.12 Using the break function 233
12.3.13 Clearing the OCxREF signal on an external event 236
12.3.14 6-step PWM generation 237
12.3.15 One-pulse mode 238
12.3.16 Encoder interface mode 239
12.3.17 Timer input XOR function 242
12.3.18 Interfacing with Hall sensors 242
12.3.19 TIMx and external trigger synchronization 244
12.3.20 Timer synchronization 247
12.3.21 Debug mode 247
12.4 TIM1 registers 248
12.4.1 TIM1 control register 1 (TIMx_CR1) 248
12.4.2 TIM1 control register 2 (TIMx_CR2) 249
12.4.3 TIM1 slave mode control register (TIMx_SMCR) 251
12.4.4 TIM1 DMA/interrupt enable register (TIMx_DIER) 253
12.4.5 TIM1 status register (TIMx_SR) 255
12.4.6 TIM1 event generation register (TIMx_EGR) 256
12.4.7 TIM1 capture/compare mode register 1 (TIMx_CCMR1) 258

12.4.8 TIM1 capture/compare mode register 2 (TIMx_CCMR2) 261
12.4.9 TIM1 capture/compare enable register (TIMx_CCER) 262
12.4.10 TIM1 counter (TIMx_CNT) 264
12.4.11 TIM1 prescaler (TIMx_PSC) 265
12.4.12 TIM1 auto-reload register (TIMx_ARR) 265
12.4.13 TIM1 repetition counter register (TIMx_RCR) 266
12.4.14 TIM1 capture/compare register 1 (TIMx_CCR1) 266
12.4.15 TIM1 capture/compare register 2 (TIMx_CCR2) 267
12.4.16 TIM1 capture/compare register 3 (TIMx_CCR3) 267
12.4.17 TIM1 capture/compare register 4 (TIMx_CCR4) 268
12.4.18 TIM1 break and dead-time register (TIMx_BDTR) 268
12.4.19 TIM1 DMA control register (TIMx_DCR) 270
12.4.20 TIM1 DMA address for full transfer (TIMx_DMAR) 271
12.4.21 TIM1 register map 272

13 General-purpose timers (TIM2 to TIM5) 274

13.1 TIM2 to TIM5 introduction 274
13.2 TIM2 to TIM5 main features 275
13.3 TIM2 to TIM5 functional description 276
13.3.1 Time-base unit 276
13.3.2 Counter modes 278
13.3.3 Clock selection 287
13.3.4 Capture/compare channels 290
13.3.5 Input capture mode 291
13.3.6 PWM input mode 292
13.3.7 Forced output mode 293
13.3.8 Output compare mode 294
13.3.9 PWM mode 295
13.3.10 One-pulse mode 298
13.3.11 Clearing the OCxREF signal on an external event 299
13.3.12 Encoder interface mode 300
13.3.13 Timer input XOR function 302
13.3.14 Timers and external trigger synchronization 302
13.3.15 Timer synchronization 305
13.3.16 Debug mode 310
13.4 TIMx2 to TIM5 registers 311
13.4.1 TIMx control register 1 (TIMx_CR1) 311

13.4.2 TIMx control register 2 (TIMx_CR2) . . . 313
13.4.3 TIMx slave mode control register (TIMx_SMCR) . . . 314
13.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER) . . . 316
13.4.5 TIMx status register (TIMx_SR) . . . 317
13.4.6 TIMx event generation register (TIMx_EGR) . . . 319
13.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) . . . 320
13.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2) . . . 323
13.4.9 TIMx capture/compare enable register (TIMx_CCER) . . . 324
13.4.10 TIMx counter (TIMx_CNT) . . . 325
13.4.11 TIMx prescaler (TIMx_PSC) . . . 326
13.4.12 TIMx auto-reload register (TIMx_ARR) . . . 326
13.4.13 TIMx capture/compare register 1 (TIMx_CCR1) . . . 327
13.4.14 TIMx capture/compare register 2 (TIMx_CCR2) . . . 327
13.4.15 TIMx capture/compare register 3 (TIMx_CCR3) . . . 328
13.4.16 TIMx capture/compare register 4 (TIMx_CCR4) . . . 328
13.4.17 TIMx DMA control register (TIMx_DCR) . . . 329
13.4.18 TIMx DMA address for full transfer (TIMx_DMAR) . . . 329
13.4.19 TIMx register map . . . 330

14 General-purpose timers (TIM12/13/14) . . . 333

14.1 TIM12/13/14 introduction . . . 333
14.2 TIM12/13/14 main features . . . 333
14.2.1 TIM12 main features . . . 333
14.3 TIM13/TIM14 main features . . . 334
14.4 TIM12/13/14 functional description . . . 336
14.4.1 Time-base unit . . . 336
14.4.2 Counter modes . . . 337
14.4.3 Clock selection . . . 340
14.4.4 Capture/compare channels . . . 342
14.4.5 Input capture mode . . . 343
14.4.6 PWM input mode (only for TIM12) . . . 344
14.4.7 Forced output mode . . . 345
14.4.8 Output compare mode . . . 346
14.4.9 PWM mode . . . 347
14.4.10 One-pulse mode (only for TIM12) . . . 348
14.4.11 TIM12 external trigger synchronization . . . 349
14.4.12 Timer synchronization (TIM12) . . . 352

14.4.13 Debug mode . . . 352
14.5 TIM12 registers . . . 353
14.5.1 TIM12 control register 1 (TIMx_CR1) . . . 353
14.5.2 TIM12 control register 2 (TIMx_CR2) . . . 354
14.5.3 TIM12 slave mode control register (TIMx_SMCR) . . . 355
14.5.4 TIM12 Interrupt enable register (TIMx_DIER) . . . 356
14.5.5 TIM12 status register (TIMx_SR) . . . 357
14.5.6 TIM12 event generation register (TIMx_EGR) . . . 358
14.5.7 TIM12 capture/compare mode register 1 (TIMx_CCMR1) . . . 359
14.5.8 TIM12 capture/compare enable register (TIMx_CCER) . . . 362
14.5.9 TIM12 counter (TIMx_CNT) . . . 363
14.5.10 TIM12 prescaler (TIMx_PSC) . . . 363
14.5.11 TIM12 auto-reload register (TIMx_ARR) . . . 363
14.5.12 TIM12 capture/compare register 1 (TIMx_CCR1) . . . 364
14.5.13 TIM12 capture/compare register 2 (TIMx_CCR2) . . . 364
14.5.14 TIM12 register map . . . 364
14.6 TIM13/14 registers . . . 366
14.6.1 TIM13/14 control register 1 (TIMx_CR1) . . . 366
14.6.2 TIM13/14 Interrupt enable register (TIMx_DIER) . . . 367
14.6.3 TIM13/14 status register (TIMx_SR) . . . 367
14.6.4 TIM13/14 event generation register (TIMx_EGR) . . . 368
14.6.5 TIM13/14 capture/compare mode register 1 (TIMx_CCMR1) . . . 369
14.6.6 TIM13/14 capture/compare enable register (TIMx_CCER) . . . 371
14.6.7 TIM13/14 counter (TIMx_CNT) . . . 372
14.6.8 TIM13/14 prescaler (TIMx_PSC) . . . 372
14.6.9 TIM13/14 auto-reload register (TIMx_ARR) . . . 372
14.6.10 TIM13/14 capture/compare register 1 (TIMx_CCR1) . . . 373
14.6.11 TIM13/14 register map . . . 373

15 General-purpose timers (TIM15/16/17) . . . 375
15.1 TIM15/16/17 introduction . . . 375
15.2 TIM15 main features . . . 375
15.3 TIM16 and TIM17 main features . . . 376
15.4 TIM15/16/17 functional description . . . 379
15.4.1 Time-base unit . . . 379

15.4.2 Counter modes . . . 380
15.4.3 Repetition counter . . . 383
15.4.4 Clock selection . . . 384
15.4.5 Capture/compare channels . . . 386
15.4.6 Input capture mode . . . 388
15.4.7 PWM input mode (only for TIM15) . . . 389
15.4.8 Forced output mode . . . 390
15.4.9 Output compare mode . . . 390
15.4.10 PWM mode . . . 391
15.4.11 Complementary outputs and dead-time insertion . . . 393
15.4.12 Using the break function . . . 394
15.4.13 One-pulse mode . . . 397
15.4.14 TIM15 and external trigger synchronization (only for TIM15) . . . 399
15.4.15 Timer synchronization . . . 401
15.4.16 Debug mode . . . 401
15.5 TIM15 registers . . . 401
15.5.1 TIM15 control register 1 (TIM15_CR1) . . . 402
15.5.2 TIM15 control register 2 (TIM15_CR2) . . . 403
15.5.3 TIM15 slave mode control register (TIM15_SMCR) . . . 404
15.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER) . . . 406
15.5.5 TIM15 status register (TIM15_SR) . . . 407
15.5.6 TIM15 event generation register (TIM15_EGR) . . . 408
15.5.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) . . . 409
15.5.8 TIM15 capture/compare enable register (TIM15_CCER) . . . 412
15.5.9 TIM15 counter (TIM15_CNT) . . . 415
15.5.10 TIM15 prescaler (TIM15_PSC) . . . 415
15.5.11 TIM15 auto-reload register (TIM15_ARR) . . . 415
15.5.12 TIM15 repetition counter register (TIM15_RCR) . . . 416
15.5.13 TIM15 capture/compare register 1 (TIM15_CCR1) . . . 416
15.5.14 TIM15 capture/compare register 2 (TIM15_CCR2) . . . 417
15.5.15 TIM15 break and dead-time register (TIM15_BDTR) . . . 417
15.5.16 TIM15 DMA control register (TIM15_DCR) . . . 419
15.5.17 TIM15 DMA address for full transfer (TIM15_DMAR) . . . 419
15.5.18 TIM15 register map . . . 420
15.6 TIM16&TIM17 registers . . . 422
15.6.1 TIM16&TIM17 control register 1 (TIMx_CR1) . . . 422
15.6.2 TIM16&TIM17 control register 2 (TIMx_CR2) . . . 423

15.6.3 TIM16&TIM17 DMA/interrupt enable register (TIMx_DIER) 424
15.6.4 TIM16&TIM17 status register (TIMx_SR) 425
15.6.5 TIM16&TIM17 event generation register (TIMx_EGR) 426
15.6.6 TIM16&TIM17 capture/compare mode register 1 (TIMx_CCMR1) . . . 427
15.6.7 TIM16&TIM17 capture/compare enable register (TIMx_CCER) 430
15.6.8 TIM16&TIM17 counter (TIMx_CNT) 432
15.6.9 TIM16&TIM17 prescaler (TIMx_PSC) 432
15.6.10 TIM16&TIM17 auto-reload register (TIMx_ARR) 432
15.6.11 TIM16&TIM17 repetition counter register (TIMx_RCR) 433
15.6.12 TIM16&TIM17 capture/compare register 1 (TIMx_CCR1) 433
15.6.13 TIM16&TIM17 break and dead-time register (TIMx_BDTR) 434
15.6.14 TIM16&TIM17 DMA control register (TIMx_DCR) 435
15.6.15 TIM16&TIM17 DMA address for full transfer (TIMx_DMAR) 436
15.6.16 TIM16&TIM17 register map 438

16 Basic timers (TIM6&TIM7) 440

16.1 TIM6&TIM7 introduction 440
16.2 TIM6&TIM7 main features 440
16.3 TIM6&TIM7 functional description 441
16.3.1 Time-base unit 441
16.3.2 Counting mode 443
16.3.3 Clock source 445
16.3.4 Debug mode 446
16.4 TIM6&TIM7 registers 446
16.4.1 TIM6&TIM7 control register 1 (TIMx_CR1) 446
16.4.2 TIM6&TIM7 control register 2 (TIMx_CR2) 448
16.4.3 TIM6&TIM7 DMA/Interrupt enable register (TIMx_DIER) 448
16.4.4 TIM6&TIM7 status register (TIMx_SR) 449
16.4.5 TIM6&TIM7 event generation register (TIMx_EGR) 449
16.4.6 TIM6&TIM7 counter (TIMx_CNT) 449
16.4.7 TIM6&TIM7 prescaler (TIMx_PSC) 450
16.4.8 TIM6&TIM7 auto-reload register (TIMx_ARR) 450
16.4.9 TIM6&TIM7 register map 451

17 Real-time clock (RTC) 452

17.1 RTC introduction 452
17.2 RTC main features 453

17.3 RTC functional description 454
17.3.1 Overview 454
17.3.2 Resetting RTC registers 455
17.3.3 Reading RTC registers 455
17.3.4 Configuring RTC registers 455
17.3.5 RTC flag assertion 456
17.4 RTC registers 457
17.4.1 RTC control register high (RTC_CRH) 457
17.4.2 RTC control register low (RTC_CRL) 458
17.4.3 RTC prescaler load register (RTC_PRLH / RTC_PRLL) 459
17.4.4 RTC prescaler divider register (RTC_DIVH / RTC_DIVL) 460
17.4.5 RTC counter register (RTC_CNTH / RTC_CNTL) 461
17.4.6 RTC alarm register high (RTC_ALRH / RTC_ALRL) 462
17.4.7 RTC register map 463

18 Independent watchdog (IWDG) 464
18.1 IWDG introduction 464
18.2 IWDG main features 464
18.3 IWDG functional description 464
18.3.1 Hardware watchdog 465
18.3.2 Register access protection 465
18.3.3 Debug mode 465
18.4 IWDG registers 466
18.4.1 Key register (IWDG_KR) 466
18.4.2 Prescaler register (IWDG_PR) 467
18.4.3 Reload register (IWDG_RLR) 467
18.4.4 Status register (IWDG_SR) 468
18.4.5 IWDG register map 468

19 Window watchdog (WWDG) 469
19.1 WWDG introduction 469
19.2 WWDG main features 469
19.3 WWDG functional description 469
19.4 How to program the watchdog timeout 471
19.5 Debug mode 472
19.6 WWDG registers 473

19.6.1 Control register (WWDG_CR) 473
19.6.2 Configuration register (WWDG_CFR) 474
19.6.3 Status register (WWDG_SR) 474
19.6.4 WWDG register map 475

20 Flexible static memory controller (FSMC) 476
20.1 FSMC main features 476
20.2 Block diagram 477
20.3 AHB interface 477
20.3.1 Supported memories and transactions 478
20.4 External device address mapping 479
20.4.1 NOR/PSRAM address mapping 479
20.5 NOR Flash/PSRAM controller 480
20.5.1 External memory interface signals 481
20.5.2 Supported memories and transactions 483
20.5.3 General timing rules 484
20.5.4 NOR Flash/PSRAM controller asynchronous transactions 484
20.5.5 Synchronous burst transactions 500
20.5.6 NOR/PSRAM controller registers 506
20.5.7 FSMC register map 512

21 Serial peripheral interface (SPI) 513
21.1 SPI introduction 513
21.2 SPI main features 514
21.2.1 SPI features 514
21.3 SPI functional description 515
21.3.1 General description 515
21.3.2 Configuring the SPI in slave mode 518
21.3.3 Configuring the SPI in master mode 520
21.3.4 Configuring the SPI for simplex communication 521
21.3.5 Data transmission and reception procedures 521
21.3.6 CRC calculation 528
21.3.7 Status flags 530
21.3.8 Disabling the SPI 531
21.3.9 SPI communication using DMA (direct memory addressing) 532
21.3.10 Error flags 533

21.3.11 SPI interrupts . . . 534
21.4 SPI registers . . . 535
21.4.1 SPI control register 1 (SPI_CR1) . . . 535
21.4.2 SPI control register 2 (SPI_CR2) . . . 536
21.4.3 SPI status register (SPI_SR) . . . 538
21.4.4 SPI data register (SPI_DR) . . . 538
21.4.5 SPI CRC polynomial register (SPI_CRCPR) . . . 539
21.4.6 SPI RX CRC register (SPI_RXCRCR) . . . 540
21.4.7 SPI TX CRC register (SPI_TXCRCR) . . . 540
21.4.8 SPI register map . . . 541

22 Inter-integrated circuit (I^2C) interface . . . 542
22.1 I^2C introduction . . . 542
22.2 I^2C main features . . . 542
22.3 I^2C functional description . . . 543
22.3.1 Mode selection . . . 543
22.3.2 I2C slave mode . . . 545
22.3.3 I2C master mode . . . 547
22.3.4 Error conditions . . . 555
22.3.5 SDA/SCL line control . . . 556
22.3.6 SMBus . . . 557
22.3.7 DMA requests . . . 559
22.3.8 Packet error checking . . . 561
22.4 I^2C interrupts . . . 561
22.5 I^2C debug mode . . . 563
22.6 I^2C registers . . . 563
22.6.1 I^2C Control register 1 (I2C_CR1) . . . 563
22.6.2 I^2C Control register 2 (I2C_CR2) . . . 565
22.6.3 I^2C Own address register 1 (I2C_OAR1) . . . 567
22.6.4 I^2C Own address register 2 (I2C_OAR2) . . . 567
22.6.5 I^2C Data register (I2C_DR) . . . 568
22.6.6 I^2C Status register 1 (I2C_SR1) . . . 568
22.6.7 I^2C Status register 2 (I2C_SR2) . . . 572
22.6.8 I^2C Clock control register (I2C_CCR) . . . 573
22.6.9 I^2C TRISE register (I2C_TRISE) . . . 574
22.6.10 I2C register map . . . 575

23 Universal synchronous asynchronous receiver transmitter (USART) 576

23.1 USART introduction 576
23.2 USART main features 576
23.3 USART functional description 577
23.3.1 USART character description 580
23.3.2 Transmitter 581
23.3.3 Receiver 584
23.3.4 Fractional baud rate generation 588
23.3.5 USART receiver's tolerance to clock deviation 590
23.3.6 Multiprocessor communication 590
23.3.7 Parity control 592
23.3.8 LIN (local interconnection network) mode 593
23.3.9 USART synchronous mode 595
23.3.10 Single-wire half-duplex communication 597
23.3.11 Smartcard 598
23.3.12 IrDA SIR ENDEC block 600
23.3.13 Continuous communication using DMA 602
23.3.14 Hardware flow control 605
23.4 USART interrupts 606
23.5 USART mode configuration 607
23.6 USART registers 607
23.6.1 Status register (USART_SR) 608
23.6.2 Data register (USART_DR) 610
23.6.3 Baud rate register (USART_BRR) 610
23.6.4 Control register 1 (USART_CR1) 611
23.6.5 Control register 2 (USART_CR2) 613
23.6.6 Control register 3 (USART_CR3) 614
23.6.7 Guard time and prescaler register (USART_GTPR) 616
23.6.8 USART register map 617

24 High-definition multimedia interface-consumer electronics control controller (HDMI™-CEC) 618

24.1 Introduction 618
24.2 HDMI-CEC main features 619
24.3 HDMI-CEC bus topology 619

24.3.1 HDMI-CEC pin . . . 619
24.3.2 Message description . . . 620
24.3.3 Bit timing . . . 621
24.4 Arbitration . . . 622
24.4.1 Signal free time (SFT) . . . 622
24.4.2 Header arbitration . . . 622
24.5 Error handling . . . 623
24.5.1 BTE, BPE and Error bit generation . . . 623
24.5.2 Message error . . . 623
24.6 Device addressing . . . 623
24.7 HDMI-CEC functional description . . . 624
24.7.1 Block diagram . . . 624
24.7.2 Prescaler . . . 624
24.7.3 Rx digital filter . . . 625
24.7.4 Rx bit timing . . . 625
24.7.5 Tx bit timing . . . 626
24.7.6 CEC arbiter . . . 627
24.7.7 CEC states . . . 628
24.7.8 CEC and system Stop mode . . . 632
24.8 HDMI-CEC interrupts . . . 633
24.9 HDMI-CEC registers . . . 634
24.9.1 CEC configuration register (CEC_CFGR) . . . 634
24.9.2 CEC own address register (CEC_OAR) . . . 635
24.9.3 CEC prescaler register (CEC_PRES) . . . 635
24.9.4 CEC error status register (CEC_ESR) . . . 636
24.9.5 CEC control and status register (CEC_CSR) . . . 637
24.9.6 CEC Tx data register (CEC_TXD) . . . 638
24.9.7 CEC Rx data register (CEC_RXD) . . . 638
24.9.8 HDMI-CEC register map . . . 639

25 Debug support (DBG) . . . 640
25.1 Overview . . . 640
25.2 Reference ARM documentation . . . 642
25.3 SWJ debug port (serial wire and JTAG) . . . 642
25.3.1 Mechanism to select the JTAG-DP or the SW-DP . . . 642
25.4 Pinout and debug port pins . . . 643

25.4.1 SWJ debug port pins . . . 644
25.4.2 Flexible SWJ-DP pin assignment . . . 644
25.4.3 Internal pull-up and pull-down on JTAG pins . . . 645
25.4.4 Using serial wire and releasing the unused debug pins as GPIOs . . . 646
25.5 STM32F100xx JTAG TAP connection . . . 646
25.6 ID codes and locking mechanism . . . 647
25.6.1 MCU device ID code . . . 647
25.6.2 Boundary scan TAP . . . 648
25.6.3 Cortex™-M3 TAP . . . 648
25.6.4 Cortex™-M3 JEDEC-106 ID code . . . 648
25.7 JTAG debug port . . . 648
25.8 SW debug port . . . 650
25.8.1 SW protocol introduction . . . 650
25.8.2 SW protocol sequence . . . 650
25.8.3 SW-DP state machine (reset, idle states, ID code) . . . 651
25.8.4 DP and AP read/write accesses . . . 652
25.8.5 SW-DP registers . . . 652
25.8.6 SW-AP registers . . . 653
25.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP . . . 653
25.10 Core debug . . . 654
25.11 Capability of the debugger host to connect under system reset . . . 655
25.12 FPB (Flash patch breakpoint) . . . 655
25.13 DWT (data watchpoint trigger) . . . 656
25.14 ITM (instrumentation trace macrocell) . . . 656
25.14.1 General description . . . 656
25.14.2 Time stamp packets, synchronization and overflow packets . . . 656
25.15 MCU debug component (DBGMCU) . . . 658
25.15.1 Debug support for low-power modes . . . 658
25.15.2 Debug support for timers, watchdog and I^2C . . . 658
25.15.3 Debug MCU configuration register . . . 659
25.16 TPIU (trace port interface unit) . . . 661
25.16.1 Introduction . . . 661
25.16.2 TRACE pin assignment . . . 661
25.16.3 TPUI formatter . . . 663
25.16.4 TPUI frame synchronization packets . . . 663

25.16.5 Transmission of the synchronization frame packet 664
25.16.6 Synchronous mode 664
25.16.7 Asynchronous mode 664
25.16.8 TRACECLKIN connection inside the STM32F100xx 664
25.16.9 TPIU registers 666
25.16.10 Example of configuration 667
25.17 DBG register map 667

26 Device electronic signature 668
26.1 Memory size registers 668
26.1.1 Flash size register 668
26.2 Unique device ID register (96 bits) 669

27 Revision history 673

List of tables

Table 1. Low and medium-density device register boundary addresses . . . 36
Table 2. High-density device register boundary addresses . . . 37
Table 3. Flash module organization (low-density value line devices) . . . 41
Table 4. Flash module organization (medium-density value line devices) . . . 42
Table 5. Flash module organization (high-density value line devices) . . . 42
Table 6. Boot modes . . . 44
Table 7. CRC calculation unit register map and reset values . . . 48
Table 8. Low-power mode summary . . . 53
Table 9. Sleep-now . . . 55
Table 10. Sleep-on-exit . . . 55
Table 11. Stop mode . . . 56
Table 12. Standby mode . . . 57
Table 13. PWR register map and reset values . . . 61
Table 14. BKP register map and reset values . . . 66
Table 15. RCC register map and reset values . . . 98
Table 16. Port bit configuration table . . . 101
Table 17. Output MODE bits . . . 101
Table 18. Advanced timer TIM1 . . . 106
Table 19. General-purpose timers TIM2/3/4/5 . . . 106
Table 20. General-purpose timers TIM15/16/17 . . . 106
Table 21. General-purpose timers TIM12/13/14 . . . 107
Table 22. USARTs . . . 107
Table 23. SPI . . . 107
Table 24. CEC . . . 108
Table 25. I2C . . . 108
Table 26. FSMC . . . 108
Table 27. Other IOs . . . 109
Table 28. Debug interface signals . . . 115
Table 29. Debug port mapping . . . 115
Table 30. TIM5 alternate function remapping . . . 116
Table 31. TIM12 remapping . . . 116
Table 32. TIM13 remapping . . . 116
Table 33. TIM14 remapping . . . 116
Table 34. TIM4 alternate function remapping . . . 116
Table 35. TIM3 alternate function remapping . . . 116
Table 36. TIM2 alternate function remapping . . . 117
Table 37. TIM1 alternate function remapping . . . 117
Table 38. TIM1 DMA remapping . . . 117
Table 39. TIM15 remapping . . . 117
Table 40. TIM16 remapping . . . 118
Table 41. TIM17 remapping . . . 118
Table 42. USART3 remapping . . . 118
Table 43. USART2 remapping . . . 118
Table 44. USART1 remapping . . . 118
Table 45. I2C1 remapping . . . 119
Table 46. SPI1 remapping . . . 119
Table 47. CEC remapping . . . 119
Table 48. GPIO register map and reset values . . . 127

Table 49. AFIO register map and reset values . . . 127
Table 50. Vector table for STM32F100xx devices . . . 129
Table 51. External interrupt/event controller register map and reset values. . . . 139
Table 52. Programmable data width & endian behavior (when bits PINC = MINC = 1) . . . 145
Table 53. DMA interrupt requests . . . 146
Table 54. Summary of DMA1 requests for each channel . . . 149
Table 55. Summary of DMA2 requests for each channel . . . 150
Table 56. DMA register map and reset values . . . 156
Table 57. ADC pins. . . . 160
Table 58. Analog watchdog channel selection . . . 163
Table 59. External trigger for regular channels for ADC1 . . . 167
Table 60. External trigger for injected channels for ADC1 . . . 167
Table 61. ADC interrupts . . . 170
Table 62. ADC register map and reset values . . . 184
Table 63. DAC pins. . . . 187
Table 64. External triggers . . . 190
Table 65. DAC register map . . . 206
Table 66. Counting direction versus encoder signals . . . 240
Table 67. TIMx Internal trigger connection . . . 253
Table 68. Output control bits for complementary OCx and OCxN channels with break feature. . . . 264
Table 69. TIM1 register map and reset values . . . 272
Table 70. Counting direction versus encoder signals . . . 301
Table 71. TIMx internal trigger connection . . . 315
Table 72. Output control bit for standard OCx channels. . . . 325
Table 73. TIMx register map and reset values . . . 330
Table 74. TIMx Internal trigger connection . . . 356
Table 75. Output control bit for standard OCx channels. . . . 363
Table 76. TIM12 register map and reset values . . . 364
Table 77. Output control bit for standard OCx channels. . . . 371
Table 78. TIM13/14 register map and reset values . . . 373
Table 79. TIMx Internal trigger connection . . . 405
Table 80. Output control bits for complementary OCx and OCxN channels with break feature . . . 414
Table 81. TIM15 register map and reset values . . . 420
Table 82. Output control bits for complementary OCx and OCxN channels with break feature . . . 431
Table 83. TIM16&TIM17 register map and reset values. . . . 438
Table 84. TIM6&TIM7 register map and reset values. . . . 451
Table 85. RTC register map and reset values . . . 463
Table 86. Min/max IWDG timeout period at 40 kHz (LSI) . . . 465
Table 87. IWDG register map and reset values . . . 468
Table 88. WWDG register map and reset values . . . 475
Table 89. NOR/PSRAM bank selection . . . 479
Table 90. External memory address. . . . 479
Table 91. Programmable NOR/PSRAM access parameters . . . 480
Table 92. Nonmultipled I/O NOR Flash . . . 481
Table 93. Multiplexed I/O NOR Flash . . . 481
Table 94. Nonmultiplexed I/Os PSRAM/SRAM . . . 482
Table 95. Multiplexed I/O PSRAM . . . 482
Table 96. NOR Flash/PSRAM supported memories and transactions . . . 483
Table 97. FSMC_BCRx bit fields . . . 486
Table 98. FSMC_BTRx bit fields . . . 486
Table 99. FSMC_BCRx bit fields . . . 488

Table 100. FSMC_BTRx bit fields . . . 488
Table 101. FSMC_BWTRx bit fields . . . 488
Table 102. FSMC_BCRx bit fields . . . 491
Table 103. FSMC_BTRx bit fields . . . 491
Table 104. FSMC_BWTRx bit fields . . . 491
Table 105. FSMC_BCRx bit fields . . . 493
Table 106. FSMC_BTRx bit fields . . . 493
Table 107. FSMC_BWTRx bit fields . . . 493
Table 108. FSMC_BCRx bit fields . . . 495
Table 109. FSMC_BTRx bit fields . . . 495
Table 110. FSMC_BWTRx bit fields . . . 495
Table 111. FSMC_BCRx bit fields . . . 497
Table 112. FSMC_BTRx bit fields . . . 497
Table 113. FSMC_BCRx bit fields . . . 502
Table 114. FSMC_BTRx bit fields . . . 503
Table 115. FSMC_BCRx bit fields . . . 505
Table 116. FSMC_BTRx bit fields . . . 505
Table 117. FSMC register map . . . 512
Table 118. SPI interrupt requests . . . 534
Table 119. SPI register map and reset values . . . 541
Table 120. SMBus vs. I2C . . . 557
Table 121. I2C Interrupt requests . . . 561
Table 122. I2C register map and reset values . . . 575
Table 123. Noise detection from sampled data . . . 587
Table 124. Error calculation for programmed baud rates . . . 589
Table 125. USART receiver's tolerance when DIV_Fraction is 0 . . . 590
Table 126. USART receiver's tolerance when DIV_Fraction is different from 0 . . . 590
Table 127. Frame formats . . . 592
Table 128. USART interrupt requests . . . 606
Table 129. USART mode configuration . . . 607
Table 130. USART register map and reset values . . . 617
Table 131. HDMI pin . . . 620
Table 132. Signal free time definition . . . 622
Table 133. Bit status depending on the low bit duration (LBD) . . . 626
Table 134. Bit status depending on the total bit duration (TBD) . . . 626
Table 135. STM32 CEC arbitration . . . 627
Table 136. Software sequence to respect when receiving a message . . . 629
Table 137. Software sequence to respect when transmitting a message . . . 630
Table 138. Software sequence to respect when transmitting a message . . . 632
Table 139. HDMI-CEC interrupts . . . 633
Table 140. HDMI-CEC register map and reset values . . . 639
Table 141. SWJ debug port pins . . . 644
Table 142. Flexible SWJ-DP pin assignment . . . 645
Table 143. JTAG debug port data registers . . . 649
Table 144. 32-bit debug port registers addressed through the shifted value A[3:2] . . . 650
Table 145. Packet request (8-bits) . . . 651
Table 146. ACK response (3 bits) . . . 651
Table 147. DATA transfer (33 bits) . . . 651
Table 148. SW-DP registers . . . 652
Table 149. Cortex™-M3 AHB-AP registers . . . 654
Table 150. Core debug registers . . . 654
Table 151. Main ITM registers . . . 657

Table 152. Asynchronous TRACE pin assignment 661
Table 153. Synchronous TRACE pin assignment 662
Table 154. Flexible TRACE pin assignment 662
Table 155. Important TPIU registers 666
Table 156. Value DBG register map and reset values 667
Table 157. Document revision history 673

List of figures

Figure 1. Low and medium density value line system architecture . . . 33
Figure 2. High density value line system architecture . . . 34
Figure 3. CRC calculation unit block diagram . . . 46
Figure 4. Power supply overview . . . 49
Figure 5. Power on reset/power down reset waveform . . . 52
Figure 6. PVD thresholds . . . 52
Figure 7. Simplified diagram of the reset circuit . . . 69
Figure 8. STM32F100xx clock tree (low and medium-density devices) . . . 70
Figure 9. STM32F100xx clock tree (high-density devices) . . . 71
Figure 10. HSE/ LSE clock sources . . . 72
Figure 11. Basic structure of a standard I/O port bit . . . 100
Figure 12. Basic structure of a five-volt tolerant I/O port bit . . . 100
Figure 13. Input floating/pull up/pull down configurations . . . 103
Figure 14. Output configuration . . . 104
Figure 15. Alternate function configuration . . . 105
Figure 16. High impedance-analog configuration . . . 106
Figure 17. ADC / DAC . . . 108
Figure 18. External interrupt/event controller block diagram . . . 133
Figure 19. External interrupt/event GPIO mapping . . . 135
Figure 20. DMA block diagram in low and medium-density STM32F100xx devices . . . 141
Figure 21. DMA block diagram in high-density STM32F100xx devices . . . 142
Figure 22. DMA1 request mapping . . . 147
Figure 23. DMA2 request mapping . . . 150
Figure 24. Single ADC block diagram . . . 159
Figure 25. Timing diagram . . . 162
Figure 26. Analog watchdog guarded area . . . 163
Figure 27. Injected conversion latency . . . 164
Figure 28. Calibration timing diagram . . . 166
Figure 29. Right alignment of data . . . 166
Figure 30. Left alignment of data . . . 166
Figure 31. Temperature sensor and VREFINT channel block diagram . . . 169
Figure 32. DAC channel block diagram . . . 187
Figure 33. Data registers in single DAC channel mode . . . 188
Figure 34. Data registers in dual DAC channel mode . . . 189
Figure 35. Timing diagram for conversion with trigger disabled TEN = 0 . . . 189
Figure 36. DAC LFSR register calculation algorithm . . . 191
Figure 37. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 192
Figure 38. DAC triangle wave generation . . . 192
Figure 39. DAC conversion (SW trigger enabled) with triangle wave generation . . . 193
Figure 40. Advanced-control timer block diagram . . . 209
Figure 41. Counter timing diagram with prescaler division change from 1 to 2 . . . 211
Figure 42. Counter timing diagram with prescaler division change from 1 to 4 . . . 211
Figure 43. Counter timing diagram, internal clock divided by 1 . . . 212
Figure 44. Counter timing diagram, internal clock divided by 2 . . . 212
Figure 45. Counter timing diagram, internal clock divided by 4 . . . 213
Figure 46. Counter timing diagram, internal clock divided by N . . . 213
Figure 47. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . 213
Figure 48. Counter timing diagram, update event when ARPE=1

(TIMx_ARR preloaded) . . . 214
Figure 49. Counter timing diagram, internal clock divided by 1 . . . 215
Figure 50. Counter timing diagram, internal clock divided by 2 . . . 215
Figure 51. Counter timing diagram, internal clock divided by 4 . . . 215
Figure 52. Counter timing diagram, internal clock divided by N . . . 216
Figure 53. Counter timing diagram, update event when repetition counter is not used . . . 216
Figure 54. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 217
Figure 55. Counter timing diagram, internal clock divided by 2 . . . 218
Figure 56. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . 218
Figure 57. Counter timing diagram, internal clock divided by N . . . 218
Figure 58. Counter timing diagram, update event with ARPE=1 (counter underflow) . . . 219
Figure 59. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . 219
Figure 60. Update rate examples depending on mode and TIMx_RCR register settings . . . 220
Figure 61. Control circuit in normal mode, internal clock divided by 1 . . . 221
Figure 62. TI2 external clock connection example . . . 221
Figure 63. Control circuit in external clock mode 1 . . . 222
Figure 64. External trigger input block . . . 222
Figure 65. Control circuit in external clock mode 2 . . . 223
Figure 66. Capture/compare channel (example: channel 1 input stage) . . . 223
Figure 67. Capture/compare channel 1 main circuit . . . 224
Figure 68. Output stage of capture/compare channel (channel 1 to 3) . . . 224
Figure 69. Output stage of capture/compare channel (channel 4) . . . 225
Figure 70. PWM input mode timing . . . 227
Figure 71. Output compare mode, toggle on OC1 . . . 229
Figure 72. Edge-aligned PWM waveforms (ARR=8) . . . 230
Figure 73. Center-aligned PWM waveforms (ARR=8) . . . 231
Figure 74. Complementary output with dead-time insertion . . . 232
Figure 75. Dead-time waveforms with delay greater than the negative pulse . . . 232
Figure 76. Dead-time waveforms with delay greater than the positive pulse . . . 233
Figure 77. Output behavior in response to a break . . . 235
Figure 78. Clearing TIMx OCxREF . . . 236
Figure 79. 6-step generation, COM example (OSSR=1) . . . 237
Figure 80. Example of one pulse mode . . . 238
Figure 81. Example of counter operation in encoder interface mode . . . 241
Figure 82. Example of encoder interface mode with TI1FP1 polarity inverted . . . 241
Figure 83. Example of hall sensor interface . . . 243
Figure 84. Control circuit in reset mode . . . 244
Figure 85. Control circuit in gated mode . . . 245
Figure 86. Control circuit in trigger mode . . . 246
Figure 87. Control circuit in external clock mode 2 + trigger mode . . . 247
Figure 88. General-purpose timer block diagram . . . 276
Figure 89. Counter timing diagram with prescaler division change from 1 to 2 . . . 277
Figure 90. Counter timing diagram with prescaler division change from 1 to 4 . . . 278
Figure 91. Counter timing diagram, internal clock divided by 1 . . . 279
Figure 92. Counter timing diagram, internal clock divided by 2 . . . 279
Figure 93. Counter timing diagram, internal clock divided by 4 . . . 279
Figure 94. Counter timing diagram, internal clock divided by N . . . 280
Figure 95. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) . . . 280
Figure 96. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) . . . 281
Figure 97. Counter timing diagram, internal clock divided by 1 . . . 282
Figure 98. Counter timing diagram, internal clock divided by 2 . . . 282

Figure 99. Counter timing diagram, internal clock divided by 4 . . . 282
Figure 100. Counter timing diagram, internal clock divided by N . . . 283
Figure 101. Counter timing diagram, Update event when repetition counter is not used . . . 283
Figure 102. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . 284
Figure 103. Counter timing diagram, internal clock divided by 2 . . . 285
Figure 104. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . 285
Figure 105. Counter timing diagram, internal clock divided by N . . . 285
Figure 106. Counter timing diagram, Update event with ARPE=1 (counter underflow) . . . 286
Figure 107. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . 286
Figure 108. Control circuit in normal mode, internal clock divided by 1 . . . 287
Figure 109. TI2 external clock connection example . . . 288
Figure 110. Control circuit in external clock mode 1 . . . 288
Figure 111. External trigger input block . . . 289
Figure 112. Control circuit in external clock mode 2 . . . 289
Figure 113. Capture/compare channel (example: channel 1 input stage) . . . 290
Figure 114. Capture/compare channel 1 main circuit . . . 290
Figure 115. Output stage of capture/compare channel (channel 1) . . . 291
Figure 116. PWM input mode timing . . . 293
Figure 117. Output compare mode, toggle on OC1 . . . 295
Figure 118. Edge-aligned PWM waveforms (ARR=8) . . . 296
Figure 119. Center-aligned PWM waveforms (ARR=8) . . . 297
Figure 120. Example of one-pulse mode . . . 298
Figure 121. Clearing TIMx OCxREF . . . 300
Figure 122. Example of counter operation in encoder interface mode . . . 301
Figure 123. Example of encoder interface mode with TI1FP1 polarity inverted . . . 302
Figure 124. Control circuit in reset mode . . . 303
Figure 125. Control circuit in gated mode . . . 304
Figure 126. Control circuit in trigger mode . . . 304
Figure 127. Control circuit in external clock mode 2 + trigger mode . . . 305
Figure 128. Master/Slave timer example . . . 306
Figure 129. Gating TIM2 with OC1REF of TIM3 . . . 307
Figure 130. Gating TIM2 with Enable of TIM3 . . . 308
Figure 131. Triggering TIM2 with update of TIM3 . . . 308
Figure 132. Triggering TIM2 with Enable of TIM3 . . . 309
Figure 133. Triggering TIM3 and TIM2 with TIM3 TI1 input . . . 310
Figure 134. General-purpose timer block diagram (TIM12) . . . 334
Figure 135. General-purpose timer block diagram (TIM13/14) . . . 335
Figure 136. Counter timing diagram with prescaler division change from 1 to 2 . . . 337
Figure 137. Counter timing diagram with prescaler division change from 1 to 4 . . . 337
Figure 138. Counter timing diagram, internal clock divided by 1 . . . 338
Figure 139. Counter timing diagram, internal clock divided by 2 . . . 338
Figure 140. Counter timing diagram, internal clock divided by 4 . . . 339
Figure 141. Counter timing diagram, internal clock divided by N . . . 339
Figure 142. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . 339
Figure 143. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . 340
Figure 144. Control circuit in normal mode, internal clock divided by 1 . . . 341
Figure 145. TI2 external clock connection example . . . 341
Figure 146. Control circuit in external clock mode 1 . . . 342
Figure 147. Capture/compare channel (example: channel 1 input stage) . . . 342
Figure 148. Capture/compare channel 1 main circuit . . . 343

Figure 149. Output stage of capture/compare channel (channel 1). 343
Figure 150. PWM input mode timing 345
Figure 151. Output compare mode, toggle on OC1. 347
Figure 152. Edge-aligned PWM waveforms (ARR=8) 348
Figure 153. Example of one pulse mode. 348
Figure 154. Control circuit in reset mode 350
Figure 155. Control circuit in gated mode 351
Figure 156. Control circuit in trigger mode. 351
Figure 157. TIM15 block diagram 377
Figure 158. TIM16 and TIM17 block diagram 378
Figure 159. Counter timing diagram with prescaler division change from 1 to 2 380
Figure 160. Counter timing diagram with prescaler division change from 1 to 4 380
Figure 161. Counter timing diagram, internal clock divided by 1 381
Figure 162. Counter timing diagram, internal clock divided by 2 381
Figure 163. Counter timing diagram, internal clock divided by 4 382
Figure 164. Counter timing diagram, internal clock divided by N. 382
Figure 165. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded). 382
Figure 166. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded). 383
Figure 167. Update rate examples depending on mode and TIMx_RCR register settings 384
Figure 168. Control circuit in normal mode, internal clock divided by 1 385
Figure 169. TI2 external clock connection example. 385
Figure 170. Control circuit in external clock mode 1 386
Figure 171. Capture/compare channel (example: channel 1 input stage) 386
Figure 172. Capture/compare channel 1 main circuit 387
Figure 173. Output stage of capture/compare channel (channel 1). 387
Figure 174. Output stage of capture/compare channel (channel 2 for TIM15) 387
Figure 175. PWM input mode timing 389
Figure 176. Output compare mode, toggle on OC1. 391
Figure 177. Edge-aligned PWM waveforms (ARR=8) 392
Figure 178. Complementary output with dead-time insertion. 393
Figure 179. Dead-time waveforms with delay greater than the negative pulse. 393
Figure 180. Dead-time waveforms with delay greater than the positive pulse. 394
Figure 181. Output behavior in response to a break.. 396
Figure 182. Example of one pulse mode. 397
Figure 183. Control circuit in reset mode 399
Figure 184. Control circuit in gated mode 400
Figure 185. Control circuit in trigger mode. 401
Figure 186. Basic timer block diagram. 441
Figure 187. Counter timing diagram with prescaler division change from 1 to 2 442
Figure 188. Counter timing diagram with prescaler division change from 1 to 4 442
Figure 189. Counter timing diagram, internal clock divided by 1 443
Figure 190. Counter timing diagram, internal clock divided by 2 444
Figure 191. Counter timing diagram, internal clock divided by 4 444
Figure 192. Counter timing diagram, internal clock divided by N. 444
Figure 193. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). 445
Figure 194. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded). 445
Figure 195. Control circuit in normal mode, internal clock divided by 1 446
Figure 196. RTC simplified block diagram 454

Figure 197. RTC second and alarm waveform example with PR=0003, ALARM=00004 456
Figure 198. RTC Overflow waveform example with PR=0003. 456
Figure 199. Independent watchdog block diagram 465
Figure 200. Watchdog block diagram 470
Figure 201. Window watchdog timing diagram 471
Figure 202. FSMC block diagram 477
Figure 203. FSMC memory banks 479
Figure 204. Mode1 read accesses. 485
Figure 205. Mode1 write accesses 485
Figure 206. ModeA read accesses 487
Figure 207. ModeA write accesses 487
Figure 208. Mode2/B read accesses 489
Figure 209. Mode2 write accesses 490
Figure 210. ModeB write accesses 490
Figure 211. ModeC read accesses 492
Figure 212. ModeC write accesses 492
Figure 213. ModeD read accesses 494
Figure 214. ModeD write accesses 494
Figure 215. Multiplexed read accesses 496
Figure 216. Multiplexed write accesses 497
Figure 217. Asynchronous wait during a read access 499
Figure 218. Asynchronous wait during a write access. 499
Figure 219. Wait configurations 501
Figure 220. Synchronous multiplexed read mode - NOR, PSRAM (CRAM) 502
Figure 221. Synchronous multiplexed write mode - PSRAM (CRAM) 504
Figure 222. SPI block diagram. 515
Figure 223. Single master/ single slave application. 516
Figure 224. Data clock timing diagram 518
Figure 225. TXE/RXNE/BSY behavior in Master / full-duplex mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers 524
Figure 226. TXE/RXNE/BSY behavior in Slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in the case of continuous transfers 524
Figure 227. TXE/BSY behavior in Master transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers 525
Figure 228. TXE/BSY in Slave transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers 526
Figure 229. RXNE behavior in receive-only mode (BIDIRMODE=0 and RXONLY=1) in the case of continuous transfers 527
Figure 230. TXE/BSY behavior when transmitting (BIDIRMODE=0 and RXONLY=0) in the case of discontinuous transfers 528
Figure 231. Transmission using DMA 532
Figure 232. Reception using DMA 533
Figure 233. I2C bus protocol 544
Figure 234. I2C block diagram 545
Figure 235. Transfer sequence diagram for slave transmitter 546
Figure 236. Transfer sequence diagram for slave receiver 547
Figure 237. Transfer sequence diagram for master transmitter 550
Figure 238. Method 1: transfer sequence diagram for master receiver 552
Figure 239. Method 2: transfer sequence diagram for master receiver when N>2 553
Figure 240. Method 2: transfer sequence diagram for master receiver when N=2 554
Figure 241. Method 2: transfer sequence diagram for master receiver when N=1 555
Figure 242. I2C interrupt mapping diagram 562

Figure 243. USART block diagram . . . 579
Figure 244. Word length programming . . . 580
Figure 245. Configurable stop bits . . . 582
Figure 246. TC/TXE behavior when transmitting . . . 583
Figure 247. Start bit detection . . . 584
Figure 248. Data sampling for noise detection . . . 586
Figure 249. Mute mode using Idle line detection . . . 591
Figure 250. Mute mode using address mark detection . . . 592
Figure 251. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 594
Figure 252. Break detection in LIN mode vs. Framing error detection . . . 595
Figure 253. USART example of synchronous transmission . . . 596
Figure 254. USART data clock timing diagram (M=0) . . . 596
Figure 255. USART data clock timing diagram (M=1) . . . 597
Figure 256. RX data setup/hold time . . . 597
Figure 257. ISO 7816-3 asynchronous protocol . . . 598
Figure 258. Parity error detection using the 1.5 stop bits . . . 600
Figure 259. IrDA SIR ENDEC- block diagram . . . 602
Figure 260. IrDA data modulation (3/16) -normal mode . . . 602
Figure 261. Transmission using DMA . . . 603
Figure 262. Reception using DMA . . . 604
Figure 263. Hardware flow control between two USARTs . . . 605
Figure 264. RTS flow control . . . 605
Figure 265. CTS flow control . . . 606
Figure 266. USART interrupt mapping diagram . . . 607
Figure 267. CEC line connection . . . 620
Figure 268. Message structure . . . 621
Figure 269. Blocks . . . 621
Figure 270. Bit timings . . . 621
Figure 271. Follower acknowledge (ACK) . . . 622
Figure 272. Signal free time . . . 622
Figure 273. Arbitration phase . . . 622
Figure 274. Error bit timing . . . 623
Figure 275. HDMI-CEC block diagram . . . 624
Figure 276. Bit timing . . . 625
Figure 277. Tx bit timing . . . 626
Figure 278. CEC control state machine . . . 628
Figure 279. Example of a complete message reception . . . 629
Figure 280. Example of a complete message transmission . . . 630
Figure 281. Example of a message transmission with transmission error . . . 631
Figure 282. CEC and system Stop mode . . . 633
Figure 283. Block diagram of STM32 MCU and Cortex™-M3-level debug support . . . 641
Figure 284. SWJ debug port . . . 642
Figure 285. JTAG TAP connections . . . 647
Figure 286. TPIU block diagram . . . 661

1 Documentation conventions

1.1 List of abbreviations for registers

The following abbreviations are used in register descriptions:

read/write (rw)	Software can read and write to these bits.
read-only (r)	Software can only read these bits.
write-only (w)	Software can only write to this bit. Reading the bit returns the reset value.
read/clear (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 (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 by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to ‘0’. Writing ‘0’ 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-only write trigger (rt_w)	Software can read this bit. Writing ‘0’ or ‘1’ triggers an event but has no effect on the bit value.
toggle (t)	Software can only toggle this bit by writing ‘1’. Writing ‘0’ has no effect.
Reserved (Res.)	Reserved bit, must be kept at reset value.

1.2 Glossary

- **Low-density value line devices** are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.
- **Medium-density value line devices** are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.
- **High-density value line devices** are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.
- **Word:** data of 32-bit length.
- **Half-word:** data of 16-bit length.
- **Byte:** data of 8-bit length.

1.3 Peripheral availability

For the availability and number of peripherals across all STM32F100xx sales types, please refer to the low, medium and high-density STM32F100xx datasheet.

2 Memory and bus architecture

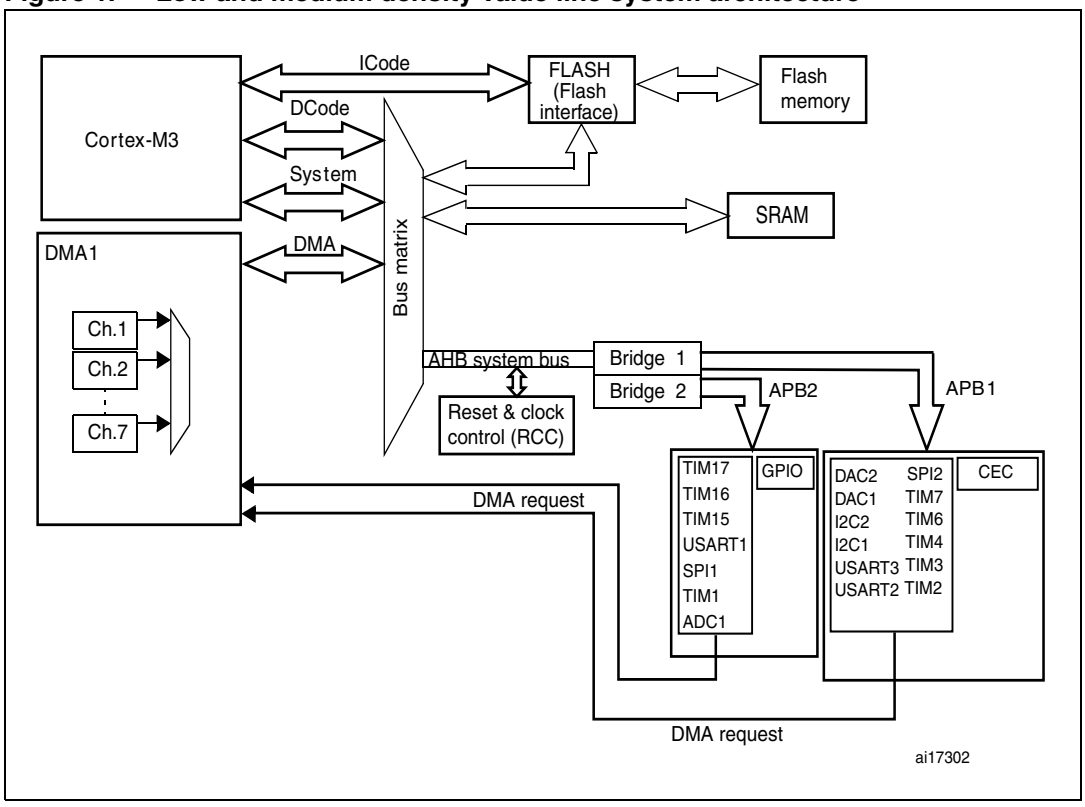
2.1 System architecture

In low-and medium-density value line devices, the main system consists of:

- Three masters:
  - Cortex™-M3 core DCode bus (D-bus) and System bus (S-bus)
  - GP-DMA1 (general-purpose DMA)
- Three slaves:
  - Internal SRAM
  - Internal Flash memory
  - AHB to APB bridges (AHB to APBx), which connect all the APB peripherals

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

Figure 1. Low and medium density value line system architecture

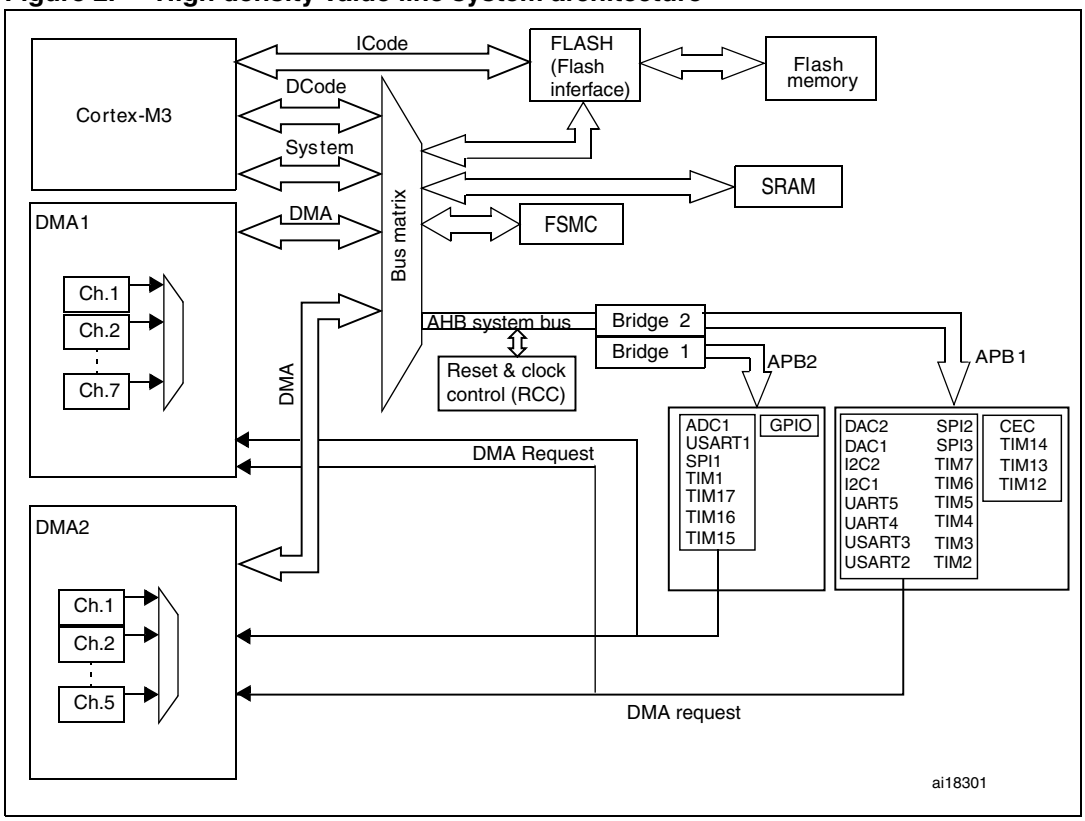


In high-density value line devices, the main system consists of:

- Four masters:
  - Cortex™-M3 core DCode bus (D-bus) and System bus (S-bus)
  - GP-DMA1 & 2 (general-purpose DMA)
- Four slaves:
  - Internal SRAM
  - Internal Flash memory
  - FSMC
  - AHB to APB bridges (AHB to APBx), which connect all the APB peripherals

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

Figure 2. High density value line system architecture



ICode bus

This bus connects the instruction bus of the Cortex™-M3 core to the Flash memory instruction interface. Instruction fetches are performed on this bus.

DCode bus

This bus connects the DCode bus (literal load and debug access) of the Cortex™-M3 core to the Flash memory data interface.

System bus

This bus connects the system bus of the Cortex™-M3 core (peripherals bus) to a bus matrix which manages the arbitration between the core and the DMA.

DMA bus

This bus connects the AHB master interface of the DMA to the bus matrix which manages the access of CPU DCode and DMA to the SRAM, Flash memory and peripherals.

Bus matrix

The bus matrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a round robin algorithm. In low and medium-density value line devices the bus matrix is composed of three masters (CPU DCode, System bus, DMA1 bus) and three slaves (FLITF, SRAM and AHB to APB bridges).

In high-density value line devices the bus matrix is composed of four masters (CPU DCode, System bus, DMA1 bus and DMA2 bus) and four slaves (FLITF, SRAM, FSMC and AHB to APB bridges).

AHB peripherals are connected to the system bus through the bus matrix to allow DMA access.

AHB/APB bridges (APB)

The two AHB/APB bridges provide full synchronous connections between the AHB and the two APB buses. APB buses operate at full speed (up to 24 MHz).

Refer to *Table 2 on page 37* for the address mapping of the peripherals connected to each bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM and FLITF). Before using a peripheral you have to enable its clock in the RCC_AHBENR, RCC_APB2ENR or RCC_APB1ENR register.

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

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.

For the detailed mapping of peripheral registers, please refer to the related sections.

The addressable memory space is divided into 8 main blocks, each of 512 MB.

All the memory areas that are not allocated to on-chip memories and peripherals are considered "reserved"). Refer to the memory map figure in the corresponding product datasheet.

2.3 Memory map

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. *Table 2* and *Table 2* give the boundary addresses of the peripherals available in all STM32F100xx devices.

Table 1. Low and medium-density device register boundary addresses

Boundary address	Peripheral	Bus	Register map
0x4002 3000 - 0x4002 33FF	CRC	AHB	*Section 3.4.4 on page 48*
0x4002 2400 - 0x4002 2FFF	Reserved		
0x4002 2000 - 0x4002 23FF	Flash memory interface		
0x4002 1400 - 0x4002 1FFF	Reserved		
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC		*Section 6.3.12 on page 98*
0x4002 0400 - 0x4002 0FFF	Reserved		
0x4002 0000 - 0x4002 03FF	DMA1		*Section 9.4.7 on page 156*
0x4001 4C00 - 0x4001 FFFF	Reserved	APB2	
0x4001 4800 - 0x4001 4BFF	TIM17 timer		*Section 15.6.16 on page 438*
0x4001 4400 - 0x4001 47FF	TIM16 timer		*Section 15.6.16 on page 438*
0x4001 4000 - 0x4001 43FF	TIM15 timer		*Section 15.5.18 on page 420*
0x4001 3C00 - 0x4001 3FFF	Reserved		
0x4001 3800 - 0x4001 3BFF	USART1		*Section 24.6.8 on page 651*
0x4001 3400 - 0x4001 37FF	Reserved		
0x4001 3000 - 0x4001 33FF	SPI1		*Section 21.4.8 on page 541*
0x4001 2C00 - 0x4001 2FFF	TIM1 timer		*Section 12.4.21 on page 272*
0x4001 2800 - 0x4001 2BFF	Reserved		
0x4001 2400 - 0x4001 27FF	ADC1		*Section 10.11.15 on page 184*
0x4001 1C00 - 0x4001 23FF	Reserved		
0x4001 1800 - 0x4001 1BFF	GPIO Port E		*Section 7.5 on page 127*
0x4001 1400 - 0x4001 17FF	GPIO Port D		*Section 7.5 on page 127*
0x4001 1000 - 0x4001 13FF	GPIO Port C		*Section 7.5 on page 127*
0x4001 0C00 - 0x4001 0FFF	GPIO Port B		*Section 7.5 on page 127*
0x4001 0800 - 0x4001 0BFF	GPIO Port A		*Section 7.5 on page 127*
0x4001 0400 - 0x4001 07FF	EXTI		*Section 8.3.7 on page 139*
0x4001 0000 - 0x4001 03FF	AFIO		*Section 7.5 on page 127*

Table 1. Low and medium-density device register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7C00 - 0x4000 FFFF	Reserved	APB1	
0x4000 7800 - 0x4000 7BFF	CEC		*Section 24.9.8 on page 639*
0x4000 7400 - 0x4000 77FF	DAC		*Section 11.5.15 on page 206*
0x4000 7000 - 0x4000 73FF	Power control PWR		*Section 4.4.3 on page 61*
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)		*Section 5.4.5 on page 66*
0x4000 5C00 - 0x4000 6BFF	Reserved		
0x4000 5800 - 0x4000 5BFF	I2C2		*Section 22.6.10 on page 575*
0x4000 5400 - 0x4000 57FF	I2C1		*Section 22.6.10 on page 575*
0x4000 4C00 - 0x4000 53FF	Reserved		
0x4000 4800 - 0x4000 4BFF	USART3		*Section 24.6.8 on page 651*
0x4000 4400 - 0x4000 47FF	USART2		*Section 24.6.8 on page 651*
0x4000 3C00 - 0x4000 3FFF	Reserved		
0x4000 3800 - 0x4000 3BFF	SPI2		*Section 21.4.8 on page 541*
0x4000 3400 - 0x4000 37FF	Reserved		
0x4000 3000 - 0x4000 33FF	Independent watchdog (IWDG)		*Section 18.4.5 on page 468*
0x4000 2C00 - 0x4000 2FFF	Window watchdog (WWDG)		*Section 19.6.4 on page 475*
0x4000 2800 - 0x4000 2BFF	RTC		*Section 17.4.7 on page 463*
0x4000 1800 - 0x4000 27FF	Reserved		
0x4000 1400 - 0x4000 17FF	TIM7 timer		*Section 16.4.9 on page 451*
0x4000 1000 - 0x4000 13FF	TIM6 timer		*Section 16.4.9 on page 451*
0x4000 0C00 - 0x4000 0FFF	Reserved		
0x4000 0800 - 0x4000 0BFF	TIM4 timer		*Section 13.4.19 on page 330*
0x4000 0400 - 0x4000 07FF	TIM3 timer		*Section 13.4.19 on page 330*
0x4000 0000 - 0x4000 03FF	TIM2 timer		*Section 13.4.19 on page 330*

Table 2. High-density device register boundary addresses

Boundary address	Peripheral	Bus	Register map
0x4002 3000 - 0x4002 33FF	CRC	AHB	*Section 3.4.4 on page 48*
0x4002 2400 - 0x4002 2FFF	Reserved		
0x4002 2000 - 0x4002 23FF	Flash memory interface		
0x4002 1400 - 0x4002 1FFF	Reserved		
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC		*Section 6.3.12 on page 98*
0x4002 0800 - 0x4002 0FFF	Reserved		
0x4002 0400 - 0x4002 07FF	DMA2		*Section 9.4.7 on page 156*
0x4002 0000 - 0x4002 03FF	DMA1		*Section 9.4.7 on page 156*

Table 2. High-density device register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4001 4C00 - 0x4001 FFFF	Reserved	APB2	
0x4001 4800 - 0x4001 4BFF	TIM17 timer		*Section 15.6.16 on page 438*
0x4001 4400 - 0x4001 47FF	TIM16 timer		*Section 15.6.16 on page 438*
0x4001 4000 - 0x4001 43FF	TIM15 timer		*Section 15.5.18 on page 420*
0x4001 3C00 - 0x4001 3FFF	Reserved		
0x4001 3800 - 0x4001 3BFF	USART1		*Section 24.6.8 on page 651*
0x4001 3400 - 0x4001 37FF	Reserved		
0x4001 3000 - 0x4001 33FF	SPI1		*Section 21.4.8 on page 541*
0x4001 2C00 - 0x4001 2FFF	TIM1 timer		*Section 12.4.21 on page 272*
0x4001 2800 - 0x4001 2BFF	Reserved		
0x4001 2400 - 0x4001 27FF	ADC1		*Section 10.11.15 on page 184*
0x4001 2000 - 0x4001 23FF	GPIO Port G		*Section 7.5 on page 127*
0x4001 1C00 - 0x4001 1FFF	GPIO Port F		*Section 7.5 on page 127*
0x4001 1800 - 0x4001 1BFF	GPIO Port E		*Section 7.5 on page 127*
0x4001 1400 - 0x4001 17FF	GPIO Port D		*Section 7.5 on page 127*
0x4001 1000 - 0x4001 13FF	GPIO Port C		*Section 7.5 on page 127*
0x4001 0C00 - 0x4001 0FFF	GPIO Port B		*Section 7.5 on page 127*
0x4001 0800 - 0x4001 0BFF	GPIO Port A		*Section 7.5 on page 127*
0x4001 0400 - 0x4001 07FF	EXTI		*Section 8.3.7 on page 139*
0x4001 0000 - 0x4001 03FF	AFIO		*Section 7.5 on page 127*

Table 2. High-density device register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7C00 - 0x4000 FFFF	Reserved	APB1	
0x4000 7800 - 0x4000 7BFF	CEC		*Section 24.9.8 on page 639*
0x4000 7400 - 0x4000 77FF	DAC		*Section 11.5.15 on page 206*
0x4000 7000 - 0x4000 73FF	Power control PWR		*Section 4.4.3 on page 61*
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)		*Section 5.4.5 on page 66*
0x4000 5C00 - 0x4000 6BFF	Reserved		
0x4000 5800 - 0x4000 5BFF	I2C2		*Section 22.6.10 on page 575*
0x4000 5400 - 0x4000 57FF	I2C1		*Section 22.6.10 on page 575*
0x4000 5000 - 0x4000 53FF	UART5		*Section 24.6.8 on page 651*
0x4000 4C00 - 0x4000 4FFF	UART4		*Section 24.6.8 on page 651*
0x4000 4800 - 0x4000 4BFF	USART3		*Section 24.6.8 on page 651*
0x4000 4400 - 0x4000 47FF	USART2		*Section 24.6.8 on page 651*
0x4000 4000 - 0x4000 43FF	Reserved		
0x4000 3C00 - 0x4000 3FFF	SPI3		*Section 21.4.8 on page 541*
0x4000 3800 - 0x4000 3BFF	SPI2		*Section 21.4.8 on page 541*
0x4000 3400 - 0x4000 37FF	Reserved		
0x4000 3000 - 0x4000 33FF	Independent watchdog (IWDG)		*Section 18.4.5 on page 468*
0x4000 2C00 - 0x4000 2FFF	Window watchdog (WWDG)		*Section 18.4.5 on page 468*
0x4000 2800 - 0x4000 2BFF	RTC		*Section 17.4.7 on page 463*
0x4000 2400 - 0x4000 27FF	Reserved		
0x4000 2000 - 0x4000 23FF	TIM14 timer		*Section 14.6.11 on page 373*
0x4000 1C00 - 0x4000 1FFF	TIM13 timer		*Section 14.6.11 on page 373*
0x4000 1800 - 0x4000 1BFF	TIM12 timer		*Section 14.5.14 on page 364*
0x4000 1400 - 0x4000 17FF	TIM7 timer		*Section 16.4.9 on page 451*
0x4000 1000 - 0x4000 13FF	TIM6 timer		*Section 16.4.9 on page 451*
0x4000 0C00 - 0x4000 0FFF	TIM5 timer		*Section 13.4.19 on page 330*
0x4000 0800 - 0x4000 0BFF	TIM4 timer		*Section 13.4.19 on page 330*
0x4000 0400 - 0x4000 07FF	TIM3 timer		*Section 13.4.19 on page 330*
0x4000 0000 - 0x4000 03FF	TIM2 timer		*Section 13.4.19 on page 330*

2.3.1 Embedded SRAM

The STM32F100xx features up to 32 Kbytes of static SRAM. It can be accessed as bytes, half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.

2.3.2 Bit banding

The Cortex™-M3 memory 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.

In the STM32F100xx, both peripheral registers and SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed.

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:

bit_word_addr = *bit_band_base* + (*byte_offset* x 32) + (*bit_number* × 4)

where:

bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit

bit_band_base is the starting address of the alias region

byte_offset is the number of the byte in the bit-band region that contains the targeted bit

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 SRAM address 0x2000 0300 in the alias region:

0x2200 6008 = 0x2200 0000 + (0x300*32) + (2*4).

Writing to address 0x2200 6008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x2000 0300.

Reading address 0x2200 6008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM address 0x2000 0300 (0x01: bit set; 0x00: bit cleared).

For more information on bit-banding, please refer to the *Cortex™-M3 Technical Reference Manual*.

2.3.3 Embedded Flash memory

The high-performance Flash memory module has the following key features:

- Density of up to 512 Kbytes
- Memory organization: the Flash memory is organized as a main block and an information block:
  - Main memory block of size:

    up to 8 Kbit × 32 bits divided into 32 pages of 1 Kbyte each for low-density value line devices (see *Table 3*)

    up to 32 Kbit × 32 bits divided into 128 pages of 1 Kbyte each for medium-density value line devices (see *Table 4*)

    up to 128 Kbit × 32 bits divided into 256 pages of 2 Kbyte each for high-density value line devices (see *Table 5*)
  - Information block of size:
    516 × 32 bits for low, medium and high-density value line devices (see *Table 3*, *Table 4* and *Table 5*)

The Flash memory interface (FLASH) features:

- Read interface (32-bit)
- Option byte loader
- Flash Program / Erase operation
- Read / write protection

Table 3. Flash module organization (low-density value line devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	. . .	. . .	. . .
	Page 31	0x0800 7C00 - 0x0800 8000	1 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Table 4. Flash module organization (medium-density value line devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbyte
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbyte
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbyte
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 Kbyte
	Page 4	0x0800 1000 - 0x0800 13FF	1 Kbyte
	. . .	. . .	. . .
	Page 127	0x0801 FC00 - 0x0801 FFFF	1 Kbyte
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Table 5. Flash module organization (high-density value line devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 07FF	2 Kbytes
	Page 1	0x0800 0800 - 0x0800 0FFF	2 Kbytes
	Page 2	0x0800 1000 - 0x0800 17FF	2 Kbytes
	Page 3	0x0800 1800 - 0x0800 1FFF	2 Kbytes
	. . .	. . .	. . .
	Page 255	0x0807 F800 - 0x0807 FFFF	2 Kbytes
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 Kbytes
	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16

Table 5. Flash module organization (high-density value line devices) (continued)

Block	Name	Base addresses	Size (bytes)
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Note: For further information on the Flash memory interface registers, please refer to the STM32F100xx Flash programming manual (PM0063).

Reading Flash memory

Flash memory accesses are performed through the AHB bus. Accesses are either instruction fetches over the ICode bus, or data accesses (e.g. literal pool) over the D-code bus. Since these two buses have the same Flash memory as target, the interface gives priority to D-code bus accesses over I-code bus, instruction fetch accesses.

Read accesses can be performed without any wait state and with the following configuration options:

- Half cycle: for power optimization

Note: 1 Half cycle configuration is not available in combination with a prescaler on the AHB. The system clock (SYSCLK) should be equal to the HCLK clock. This feature can therefore be used only with a low-frequency clock of 8 MHz or less. It can be generated from the HSI or the HSE but not from the PLL.

2 Using DMA: DMA accesses Flash memory on the DCode bus and has priority over ICode instructions. The DMA provides one free cycle after each transfer. Some instructions can be performed together with DMA transfer.

Programming and erasing Flash memory

The Flash memory can be programmed 16 bits (half words) at a time. For write and erase operations on the Flash memory (write/erase), the internal RC oscillator (HSI) must be ON.

The Flash memory erase operation can be performed at page level or on the whole Flash area (mass erase). Mass erase does not affect the information blocks.

To ensure that there is no overprogramming, the Flash programming and erase controller blocks are clocked by a fixed clock.

The end of write operation (programming or erasing) can trigger an interrupt. This interrupt can be used to exit the WFI mode, only if the FLASH clock is enabled. Otherwise, the interrupt is served only after exiting WFI.

The FLASH_ACR register is used to enable/disable Flash memory half-cycle access. The tables below provide the bit map and bit descriptions for this register.

For complete information on Flash memory operations and register configurations, please refer to the STM32F100xx Flash programming manual (PM0063).

Flash access control register (FLASH_ACR)

Address offset: 0x00
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												HLFCYA	Reserved		
												rw			

Bits 31:4 Reserved, must be kept cleared.

Bit 3 **HLFCYA**: Flash half cycle access enable
0: Half cycle is disabled
1: Half cycle is enabled

Bits 2:0 Reserved, must be kept cleared.

2.4 Boot configuration

In the STM32F100xx, three different boot modes can be selected through the BOOT[1:0] pins as shown in *Table 6*.

Table 6. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a reset. It is up to the application to set the BOOT1 and BOOT0 pins after reset to select the required boot mode.

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

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). The Cortex-M3 CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). STM32F100xx microcontrollers implement a special mechanism to be able to boot also from SRAM and not only from main Flash memory and System memory.

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

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF F000).
- Boot from the embedded SRAM: SRAM is accessible only at address 0x2000 0000.

Note: When booting from SRAM, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and offset register.

Embedded boot loader

The embedded boot loader is used to reprogram the Flash memory using the USART1 serial interface. This program is located in the system memory and is programmed by ST during production. For further details please refer to AN2606.

3 CRC calculation unit

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

3.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial.

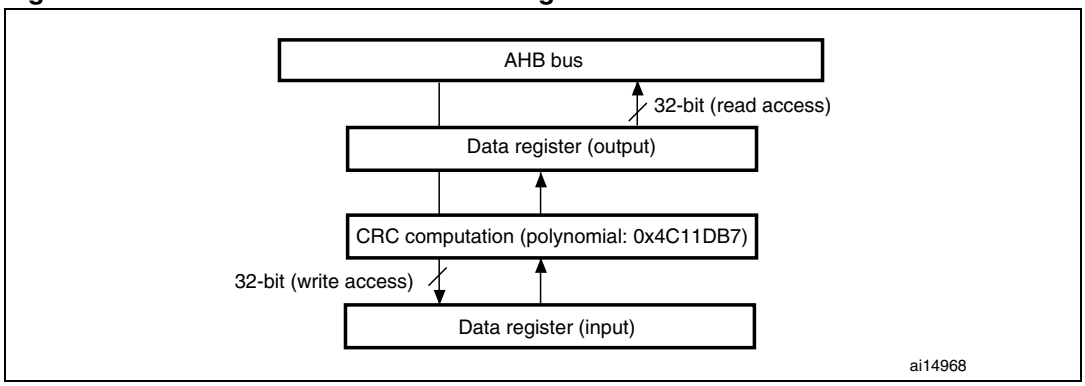
Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link-time and stored at a given memory location.

3.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Single input/output 32-bit data register
- CRC computation done in 4 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used for temporary storage)

The block diagram is shown in *Figure 3*.

Figure 3. CRC calculation unit block diagram



3.3 CRC functional description

The CRC calculation unit mainly consists of a single 32-bit data register, which:

- is used as an input register to enter new data in the CRC calculator (when writing into the register)
- holds the result of the previous CRC calculation (when reading the register)

Each write operation into the data register creates a combination of the previous CRC value and the new one (CRC computation is done on the whole 32-bit data word, and not byte per byte).

The write operation is stalled until the end of the CRC computation, thus allowing back-to-back write accesses or consecutive write and read accesses.

The CRC calculator can be reset to 0xFFFF FFFF with the RESET control bit in the CRC_CR register. This operation does not affect the contents of the CRC_IDR register.

3.4 CRC registers

The CRC calculation unit contains two data registers and a control register.

3.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR [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
DR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **Data register bits**

Used as an input register when writing new data into the CRC calculator.
Holds the previous CRC calculation result when it is read.

3.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **General-purpose 8-bit data register bits**

Can be used as a temporary storage location for one byte.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.

3.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
															w

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

Bit 0 **RESET bit**

Resets the CRC calculation unit and sets the data register to 0xFFFF FFFF.
This bit can only be set, it is automatically cleared by hardware.

3.4.4 CRC register map

The following table provides the CRC register map and reset values.

Table 7. CRC calculation unit register map and reset values

Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	**CRC_DR** Reset value	Data register 0xFFFF FFFF										
0x04	**CRC_IDR** Reset value	Reserved			Independent data register 0x00							
0x08	**CRC_CR** Reset value	Reserved										RESET 0

4 Power control (PWR)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

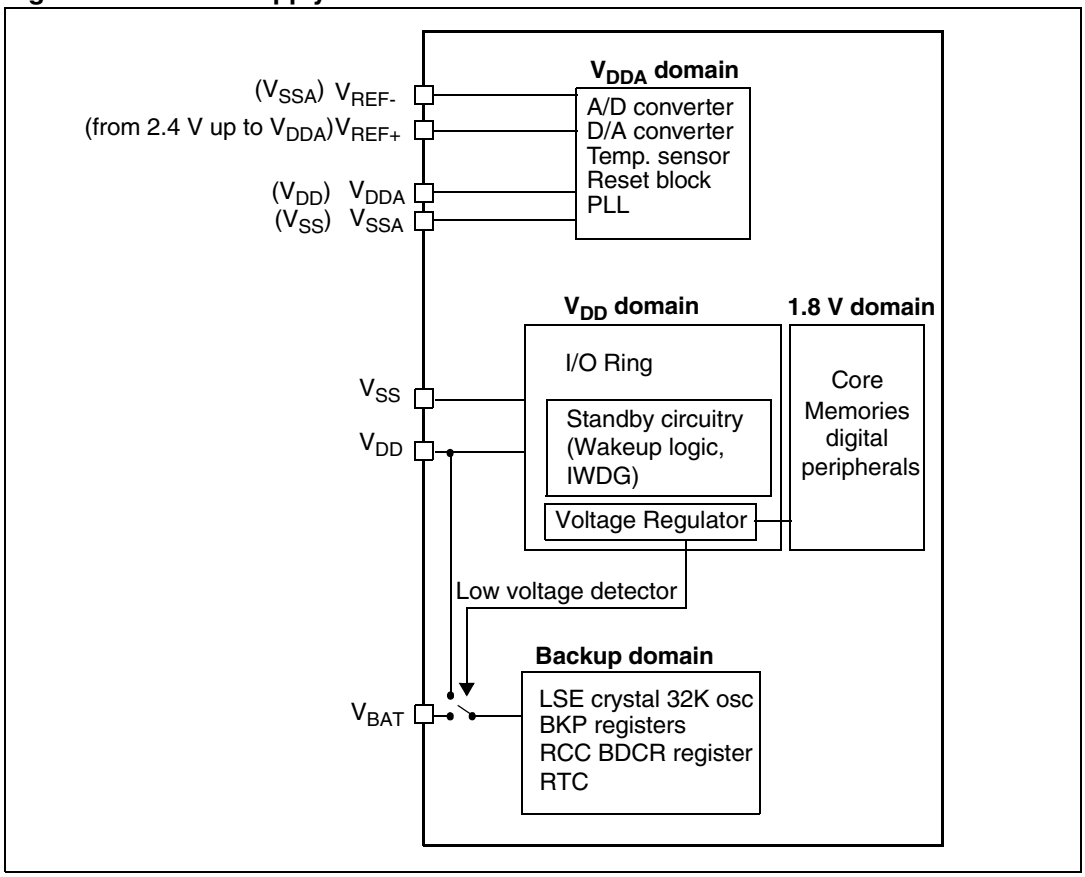
This section applies to the whole STM32F100xx family, unless otherwise specified.

4.1 Power supplies

The device requires a 2.0-to-3.6 V operating voltage supply (V_{DD}). An embedded regulator is used to supply the internal 1.8 V digital power.

The real-time clock (RTC) and backup registers can be powered from the V_{BAT} voltage when the main V_{DD} supply is powered off.

Figure 4. Power supply overview



Note: 1 V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.

4.1.1 Independent A/D and D/A converter supply and reference voltage

To improve conversion accuracy, the ADC and the DAC have an independent power supply which can be separately filtered and shielded from noise on the PCB.

- The ADC and DAC voltage supply input is available on a separate V_{DDA} pin.
- An isolated supply ground connection is provided on pin V_{SSA}.

When available (according to package), V_{REF-} must be tied to V_{SSA}.

On 100-pin packages

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect a separate external reference voltage on V_{REF+}. V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal. The voltage on V_{REF+} can range from 2.4 V to V_{DDA}.

On 64-pin packages and packages with less pins

The V_{REF+} and V_{REF-} pins are not available, they are internally connected to the ADC voltage supply (V_{DDA}) and ground (V_{SSA}).

4.1.2 Battery backup domain

To retain the content of the Backup registers and supply the RTC function when V_{DD} is turned off, V_{BAT} pin can be connected to an optional standby voltage supplied by a battery or by another source.

The V_{BAT} pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 IOs, allowing the RTC to operate even when the main digital supply (V_{DD}) is turned off. The switch to the V_{BAT} supply is controlled by the Power Down Reset embedded in the Reset block.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (Refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the V_{BAT} pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the V_{BAT} pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} with a 100 nF external ceramic decoupling capacitor (for more details refer to AN2586).

When the backup domain is supplied by V_{DD} (analog switch connected to V_{DD}), the following functions are available:

- PC14 and PC15 can be used as either GPIO or LSE pins
- PC13 can be used as GPIO, TAMPER pin, RTC Calibration Clock, RTC Alarm or second output (refer to *Section 5: Backup registers (BKP) on page 62*)

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

When the backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the following functions are available:

- PC14 and PC15 can be used as LSE pins only
- PC13 can be used as TAMPER pin, RTC Alarm or Second output (refer to section *Section 5.4.2: RTC clock calibration register (BKP_RTCCR) on page 64*).

4.1.3 Voltage regulator

The voltage regulator is always enabled after Reset. It works in three different modes depending on the application modes.

- In Run mode, the regulator supplies full power to the 1.8 V domain (core, memories and digital peripherals).
- In Stop mode the regulator supplies low-power to the 1.8 V domain, preserving contents of registers and SRAM
- In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost except for the Standby circuitry and the Backup Domain.

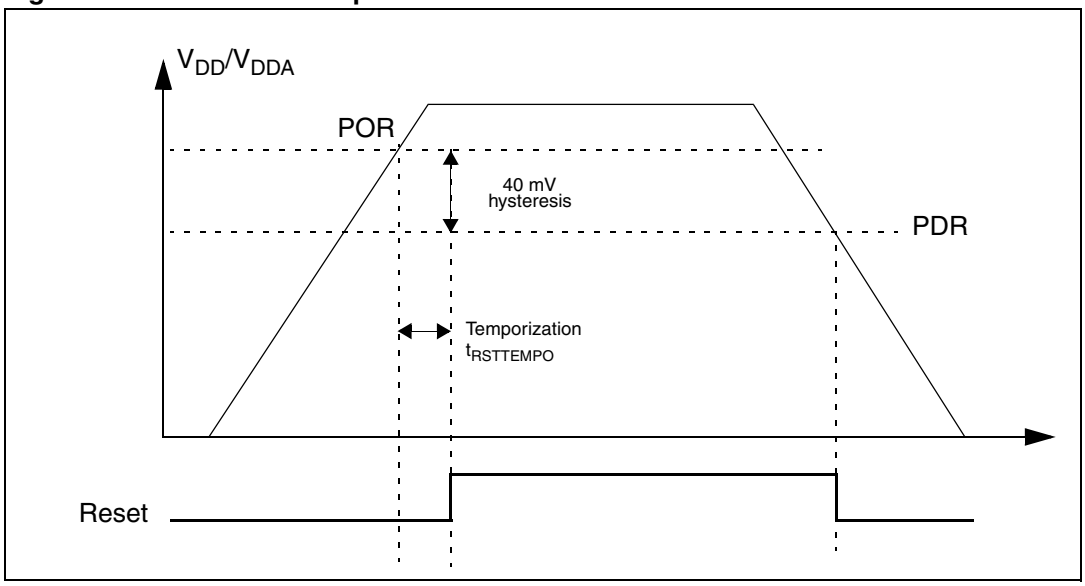
4.2 Power supply supervisor

4.2.1 Power on reset (POR)/power down reset (PDR)

The device has an integrated POR/PDR circuitry that allows proper operation starting from/down to 2 V.

The device remains in Reset mode when V_{DD}/V_{DDA} is below a specified threshold, $V_{POR/PDR}$, without the need for an external reset circuit. For more details concerning the power on/power down reset threshold, refer to the electrical characteristics of the datasheet.

Figure 5. Power on reset/power down reset waveform



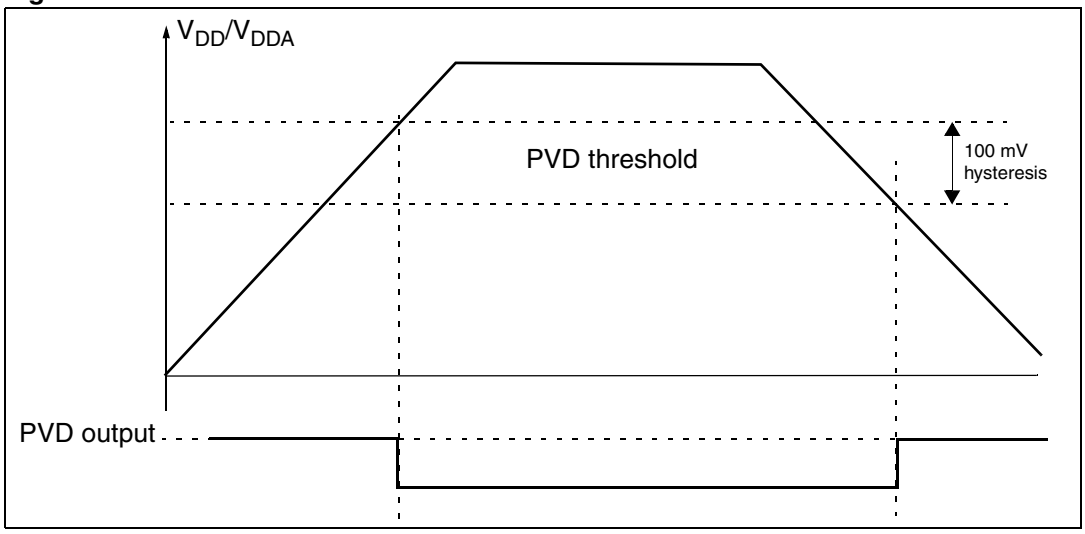
4.2.2 Programmable voltage detector (PVD)

You can use the PVD to monitor the V_{DD}/V_{DDA} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *Power control register (PWR_CR)*.

The PVD is enabled by setting the PVDE bit.

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

Figure 6. PVD thresholds



4.3 Low-power modes

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

The STM32F100xx devices feature three low-power modes:

- Sleep mode (CPU clock off, all peripherals including Cortex-M3 core peripherals like NVIC, SysTick, etc. are kept running)
- Stop mode (all clocks are stopped)
- Standby mode (1.8V domain powered-off)

In addition, the power consumption in Run mode can be reduce by one of the following means:

- Slowing down the system clocks
- Gating the clocks to the APB and AHB peripherals when they are unused.

Table 8. Low-power mode summary

Mode name	Entry	wakeup	Effect on 1.8V domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on - exit)	WFI	Any interrupt	CPU clock OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Stop	PDDS and LPDS bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers)	All 1.8V domain clocks OFF	HSI and HSE oscillators OFF	ON or in low-power mode (depends on *Power control register (PWR_CR)*)
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			OFF

4.3.1 Slowing down system clocks

In Run mode the speed of the system clocks (SYSCLK, HCLK, PCLK1, PCLK2) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down peripherals before entering Sleep mode.

For more details refer to *Section 7.3.2: Clock configuration register (RCC_CFGR)*.

4.3.2 Peripheral clock gating

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

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

Peripheral clock gating is controlled by the , *APB1 peripheral clock enable register (RCC_APB1ENR)* and *APB2 peripheral clock enable register (RCC_APB2ENR)*.

4.3.3 Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex-M3 System Control register:

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

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

Refer to *Table 9* and *Table 10* for details on how to enter Sleep mode.

Exiting Sleep mode

If the WFI instruction is used to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode.

If the WFE instruction is used to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated either by:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex-M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

This mode offers the lowest wakeup time as no time is wasted in interrupt entry/exit.

Refer to *Table 9* and *Table 10* for more details on how to exit Sleep mode.

Table 9. Sleep-now

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex™-M3 System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to *Table 50: Vector table for STM32F100xx devices* If WFE was used for entry Wakeup event: Refer to *Section 8.2.3: Wakeup event management*
Wakeup latency	None

Table 10. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex™-M3 System Control register.
Mode exit	Interrupt: refer to *Table 50: Vector table for STM32F100xx devices*.
Wakeup latency	None

4.3.4 Stop mode

The Stop mode is based on the Cortex-M3 deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the 1.8 V domain are stopped, the PLL, the HSI and the HSE RC oscillators are disabled. SRAM and register contents are preserved.

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

Entering Stop mode

Refer to *Table 11* for details on how to enter the Stop mode.

To further reduce power consumption in Stop mode, the internal voltage regulator can be put in low-power mode. This is configured by the LPDS bit of the *Power control register (PWR_CR)*.

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

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

In Stop 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 18.3: IWDG functional description* in *Section 18: Independent watchdog (IWDG)*.

- real-time clock (RTC): this is configured by the RTCEN bit in the *Backup domain control register (RCC_BDCR)*
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the *Control/status register (RCC_CSR)*.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the *Backup domain control register (RCC_BDCR)*.

The ADC or DAC can also consume power during the Stop mode, unless they are disabled before entering it. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register must both be written to 0.

Exiting Stop mode

Refer to *Table 11* for more details on how to exit Stop mode.

When exiting Stop mode by issuing an interrupt or a wakeup event, the HSI RC oscillator is selected as system clock.

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

Table 11. Stop mode

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP bit in Cortex™-M3 System Control register – Clear PDDS bit in Power Control register (PWR_CR) – Select the voltage regulator mode by configuring LPDS bit in PWR_CR **Note:** To enter Stop mode, all EXTI Line pending bits (in *Pending register (EXTI_PR)*) and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.
Mode exit	If WFI was used for entry: Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Refer to *Table 50: Vector table for STM32F100xx devices*. If WFE was used for entry: Any EXTI Line configured in event mode. Refer to *Section 8.2.3: Wakeup event management on page 133*
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

4.3.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex-M3 deepsleep mode, with the voltage regulator disabled. The 1.8 V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the Backup domain and Standby circuitry (see *Figure 4*).

Entering Standby mode

Refer to *Table 12* for more 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 18.3: IWDG functional description* in *Section 18: Independent watchdog (IWDG)*.
- real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin or the rising edge of an RTC alarm occurs (see *Figure 196: RTC simplified block diagram*). All registers are reset after wakeup from Standby except for *Power control/status register (PWR_CSR)*.

After waking up from Standby mode, program execution restarts in the same way as after a Reset (boot pins sampling, vector reset is fetched, etc.). The SBF status flag in the *Power control/status register (PWR_CSR)* indicates that the MCU was in Standby mode.

Refer to *Table 12* for more details on how to exit Standby mode.

Table 12. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in Cortex™-M3 System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR)
Mode exit	WKUP pin rising edge, RTC alarm event's rising edge, external Reset in NRST pin, IWDG Reset.
Wakeup latency	Reset phase

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance except:

- Reset pad (still available)
- TAMPER pin if configured for tamper or calibration out
- WKUP pin, if enabled

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex™-M3 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 25.15.1: Debug support for low-power modes*.

4.3.6 Auto-wakeup (AWU) from low-power mode

The RTC can be used to wakeup the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop 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 *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 (less than 1µA added consumption in typical conditions)
- Low-power internal RC Oscillator (LSI RC)
  This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.

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

4.4 Power control registers

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

4.4.1 Power control register (PWR_CR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by wakeup from Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

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

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

Note: If the HSE divided by 128 is used as the RTC clock, this bit must remain set to 1.

Bits 7:5 **PLS[2:0]:** PVD level selection.

These bits are written by software to select the voltage threshold detected by the Power Voltage Detector

000: 2.2V

001: 2.3V

010: 2.4V

011: 2.5V

100: 2.6V

101: 2.7V

110: 2.8V

111: 2.9V

Note: Refer to the electrical characteristics of the datasheet for more details.

Bit 4 **PVDE:** Power voltage detector enable.

This bit is set and cleared by software.

0: PVD disabled

1: PVD enabled

Bit 3 **CSBF**: Clear standby flag.

This bit is always read as 0.

0: No effect

1: Clear the SBF Standby Flag (write).

Bit 2 **CWUF:** Clear wakeup flag.

This bit is always read as 0.

0: No effect

1: Clear the WUF Wakeup Flag **after 2 System clock cycles**. (write)

Bit 1 **PDDS**: Power down deepsleep.

This bit is set and cleared by software. It works together with the LPDS bit.

0: Enter Stop mode when the CPU enters Deepsleep. The regulator status depends on the LPDS bit.

1: Enter Standby mode when the CPU enters Deepsleep.

Bit 0 **LPDS:** Low-power deepsleep.

This bit is set and cleared by software. It works together with the PDDS bit.

0: Voltage regulator on during Stop mode

1: Voltage regulator in low-power mode during Stop mode

4.4.2 Power control/status register (PWR_CSR)

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from Standby mode)

Additional APB cycles are needed to read this register versus a standard APB read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							EWUP	Reserved					PVDO	SBF	WUF
							rw						r	r	r

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

Bit 8 **EWUP:** Enable WKUP pin

This bit is set and cleared by software.

0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from Standby mode.

1: WKUP pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin wakes-up the system from Standby mode).

Note: This bit is reset by a system Reset.

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

Bit 2 **PVDO:** PVD output

This bit is set and cleared by hardware. It is valid only if PVD is enabled by the PVDE bit.

0: V_{DD}/V_{DDA} is higher than the PVD threshold selected with the PLS[2:0] bits.

1: V_{DD}/V_{DDA} is lower than the PVD threshold selected with the PLS[2:0] bits.

Note: The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bit 1 **SBF:** Standby flag

This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CSBF bit in the *Power control register (PWR_CR)*

0: Device has not been in Standby mode

1: Device has been in Standby mode

Bit 0 **WUF:** Wakeup flag

This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CWUF bit in the *Power control register (PWR_CR)*

0: No wakeup event occurred

1: A wakeup event was received from the WKUP pin or from the RTC alarm

Note: An additional wakeup event is detected if the WKUP pin is enabled (by setting the EWUP bit) when the WKUP pin level is already high.

4.4.3 PWR register map

The following table summarizes the PWR registers.

Table 13. 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_CR**	Reserved																							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																								0	0	0	0	0	0	0	0	0
0x004	**PWR_CSR**	Reserved																							EWUP	Reserved					PVDO	SBF	WUF
	Reset value																								0						0	0	0

Refer to *Table 1 on page 36* for the register boundary addresses.

5 Backup registers (BKP)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

5.1 BKP introduction

The backup registers are ten 16-bit registers in low and medium density devices and fort-two registers in high-density devices for storing 20 or 84 bytes of user application data.

They are implemented in the backup domain that remains powered on by V_{BAT} when the V_{DD} power is switched off. They are not reset when the device wakes up from Standby mode or by a system reset or power reset.

In addition, the BKP control registers are used to manage the Tamper detection feature and RTC calibration.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:

- enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register
- set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

5.2 BKP main features

- 20-byte data registers (in low and medium-density devices) or 84-byte data registers (in high-density devices)
- Status/control register for managing tamper detection with interrupt capability
- Calibration register for storing the RTC calibration value
- Possibility to output the RTC Calibration Clock, RTC Alarm pulse or Second pulse on TAMPER pin PC13 (when this pin is not used for tamper detection)

5.3 BKP functional description

5.3.1 Tamper detection

The TAMPER pin generates a Tamper detection event when the pin changes from 0 to 1 or from 1 to 0 depending on the TPAL bit in the *Backup control register (BKP_CR)*. A tamper detection event resets all data backup registers.

However to avoid losing Tamper events, the signal used for edge detection is logically ANDed with the Tamper enable in order to detect a Tamper event in case it occurs before the TAMPER pin is enabled.

- **When TPAL=0:** If the TAMPER pin is already high before it is enabled (by setting TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no rising edge on the TAMPER pin after TPE was set)
- **When TPAL=1:** If the TAMPER pin is already low before it is enabled (by setting the TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no falling edge on the TAMPER pin after TPE was set)

By setting the TPIE bit in the BKP_CSR register, an interrupt is generated when a Tamper detection event occurs.

After a Tamper event has been detected and cleared, the TAMPER pin should be disabled and then re-enabled with TPE before writing to the backup data registers (BKP_DRx) again. This prevents software from writing to the backup data registers (BKP_DRx), while the TAMPER pin value still indicates a Tamper detection. This is equivalent to a level detection on the TAMPER pin.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the data backup registers, the TAMPER pin should be externally tied to the correct level.

5.3.2 RTC calibration

For measurement purposes, the RTC clock with a frequency divided by 64 can be output on the TAMPER pin. This is enabled by setting the CCO bit in the *RTC clock calibration register (BKP_RTCCR)*.

The clock can be slowed down by up to 121 ppm by configuring CAL[6:0] bits.

For more details about RTC calibration and how to use it to improve timekeeping accuracy, please refer to AN2604 "*STM32F101xx and STM32F103xx RTC calibration*".

5.4 BKP registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

5.4.1 Backup data register x (BKP_DRx) (x = 1 ..42)

Address offset: 0x04 to 0x28, 0x40 to 0xBC

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **D[15:0]** Backup data

These bits can be written with user data.

Note: The BKP_DRx registers are not reset by a System reset or Power reset or when the device wakes up from Standby mode.
They are reset by a Backup Domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).

5.4.2 RTC clock calibration register (BKP_RTCCR)

Address offset: 0x2C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ASOS	ASOE	CCO	CAL[6:0]						
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ASOS:** Alarm or second output selection

When the ASOE bit is set, the ASOS bit can be used to select whether the signal output on the TAMPER pin is the RTC Second pulse signal or the Alarm pulse signal:
0: RTC Alarm pulse output selected
1: RTC Second pulse output selected

Note: This bit is reset only by a Backup domain reset.

Bit 8 **ASOE:** Alarm or second output enable

Setting this bit outputs either the RTC Alarm pulse signal or the Second pulse signal on the TAMPER pin depending on the ASOS bit.
The output pulse duration is one RTC clock period. The TAMPER pin must not be enabled while the ASOE bit is set.

Note: This bit is reset only by a Backup domain reset.

Bit 7 **CCO:** Calibration clock output

0: No effect
1: Setting this bit outputs the RTC clock with a frequency divided by 64 on the TAMPER pin. The TAMPER pin must not be enabled while the CCO bit is set in order to avoid unwanted Tamper detection.

Note: This bit is reset when the V_{DD} supply is powered off.

Bit 6:0 **CAL[6:0]:** Calibration value

This value indicates the number of clock pulses that will be ignored every 2^20 clock pulses. This allows the calibration of the RTC, slowing down the clock by steps of 1000000/2^20 PPM.

The clock of the RTC can be slowed down from 0 to 121PPM.

5.4.3 Backup control register (BKP_CR)

Address offset: 0x30

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TPAL	TPE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **TPAL:** TAMPER pin active level

0: A high level on the TAMPER pin resets all data backup registers (if TPE bit is set).
1: A low level on the TAMPER pin resets all data backup registers (if TPE bit is set).

Bit 0 **TPE:** TAMPER pin enable

0: The TAMPER pin is free for general purpose I/O
1: Tamper alternate I/O function is activated.

Note: Setting the TPAL and TPE bits at the same time is always safe, however resetting both at the same time can generate a spurious Tamper event. For this reason it is recommended to change the TPAL bit only when the TPE bit is reset.

5.4.4 Backup control/status register (BKP_CSR)

Address offset: 0x34

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIF	TEF	Reserved					TPIE	CTI	CTE
						r	r						rw	w	w

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **TIF:** Tamper interrupt flag

This bit is set by hardware when a Tamper event is detected and the TPIE bit is set. It is cleared by writing 1 to the CTI bit (also clears the interrupt). It is also cleared if the TPIE bit is reset.

0: No Tamper interrupt
1: A Tamper interrupt occurred

Note: This bit is reset only by a system reset and wakeup from Standby mode.

Bit 8 **TEF:** Tamper event flag

This bit is set by hardware when a Tamper event is detected. It is cleared by writing 1 to the CTE bit.

0: No Tamper event

1: A Tamper event occurred

Note: A Tamper event resets all the BKP_DRx registers. They are held in reset as long as the TEF bit is set. If a write to the BKP_DRx registers is performed while this bit is set, the value will not be stored.

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

Bit 2 **TPIE:** TAMPER pin interrupt enable

0: Tamper interrupt disabled

1: Tamper interrupt enabled (the TPE bit must also be set in the BKP_CR register

*Note: **1:** A Tamper interrupt does not wake up the core from low-power modes.*

__2:__ This bit is reset only by a system reset and wakeup from Standby mode.

Bit 1 **CTI:** Clear tamper interrupt

This bit is write only, and is always read as 0.

0: No effect

1: Clear the Tamper interrupt and the TIF Tamper interrupt flag.

Bit 0 **CTE:** Clear tamper event

This bit is write only, and is always read as 0.

0: No effect

1: Reset the TEF Tamper event flag (and the Tamper detector)

5.4.5 BKP register map

BKP registers are mapped as 16-bit addressable registers as described in the table below:

Table 14. BKP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	Reserved																																
0x04	**BKP_DR1**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**BKP_DR2**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**BKP_DR3**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**BKP_DR4**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**BKP_DR5**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**BKP_DR6**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14. BKP 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
0x1C	**BKP_DR7**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**BKP_DR8**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**BKP_DR9**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**BKP_DR10**	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2	**BKP_RTCCR**	Reserved																						ASOS	ASOE	CCO	CAL[6:0]						
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x30	**BKP_CR**	Reserved																														TPAL	TPE
	Reset value																															0	0
0x34	**BKP_CSR**	Reserved																						TIF	TEF	Reserved					TPIE	CTI	CTE
	Reset value																							0	0						0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

6 Reset and clock control (RCC)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to all STM32F100xx devices, unless otherwise specified.

6.1 Reset

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

6.1.1 System reset

A system reset sets all registers to their reset values except the reset flags in the clock controller CSR register and the registers in the Backup domain (see *Figure 4*).

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 end of count condition (WWDG reset)
3. Independent watchdog end of count condition (IWDG reset)
4. A software reset (SW reset) (see *Software reset*)
5. Low-power management reset (see *Low-power management reset*)

The reset source can be identified by checking the reset flags in the Control/Status register, RCC_CSR (see *Section 6.3.10: Control/status register (RCC_CSR)*).

Software reset

The SYSRESETREQ bit in Cortex™-M3 Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the Cortex™-M3 technical reference manual for more details.

Low-power management reset

There are two ways to generate a low-power management reset:

1. Reset generated when entering Standby mode:

   This type of reset is enabled by resetting nRST_STDBY bit in User Option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
2. Reset when entering Stop mode:

   This type of reset is enabled by resetting NRST_STOP bit in User Option Bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

For further information on the User Option Bytes, refer to the STM32F100xx Flash programming manual.

6.1.2 Power reset

A power reset is generated when one of the following events occurs:

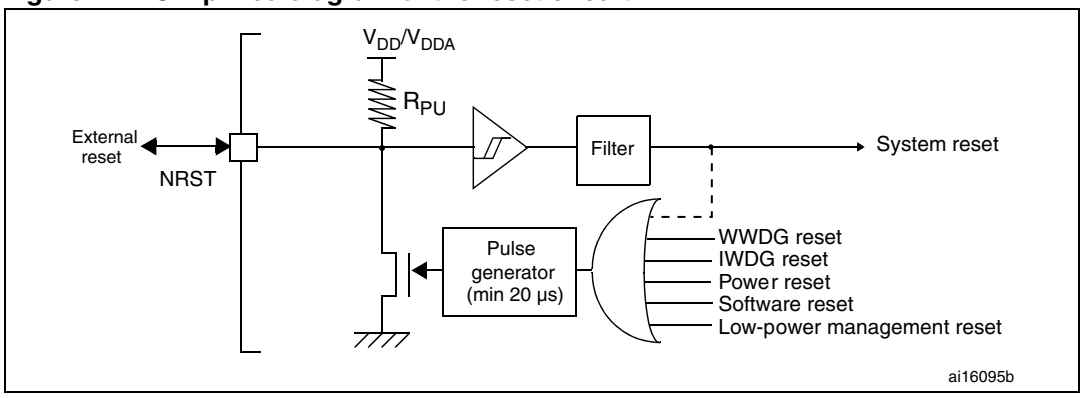
1. Power-on/power-down reset (POR/PDR reset)
2. When exiting Standby mode

A power reset sets all registers to their reset values except the Backup domain (see *Figure 4*)

These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map. For more details, refer to *Table 51: External interrupt/event controller register map and reset values on page 139*.

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 reset source (external or internal reset). In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Figure 7. Simplified diagram of the reset circuit



6.1.3 Backup domain reset

The backup domain has two specific resets that affect only the backup domain (see *Figure 4*).

A backup domain reset is generated when one of the following events occurs:

1. Software reset, triggered by setting the BDRST bit in the *Backup domain control register (RCC_BDCR)*.
2. V_{DD} or V_{BAT} power on, if both supplies have previously been powered off.

6.2 Clocks

Three different clock sources can be used to drive the system clock (SYSCLK):

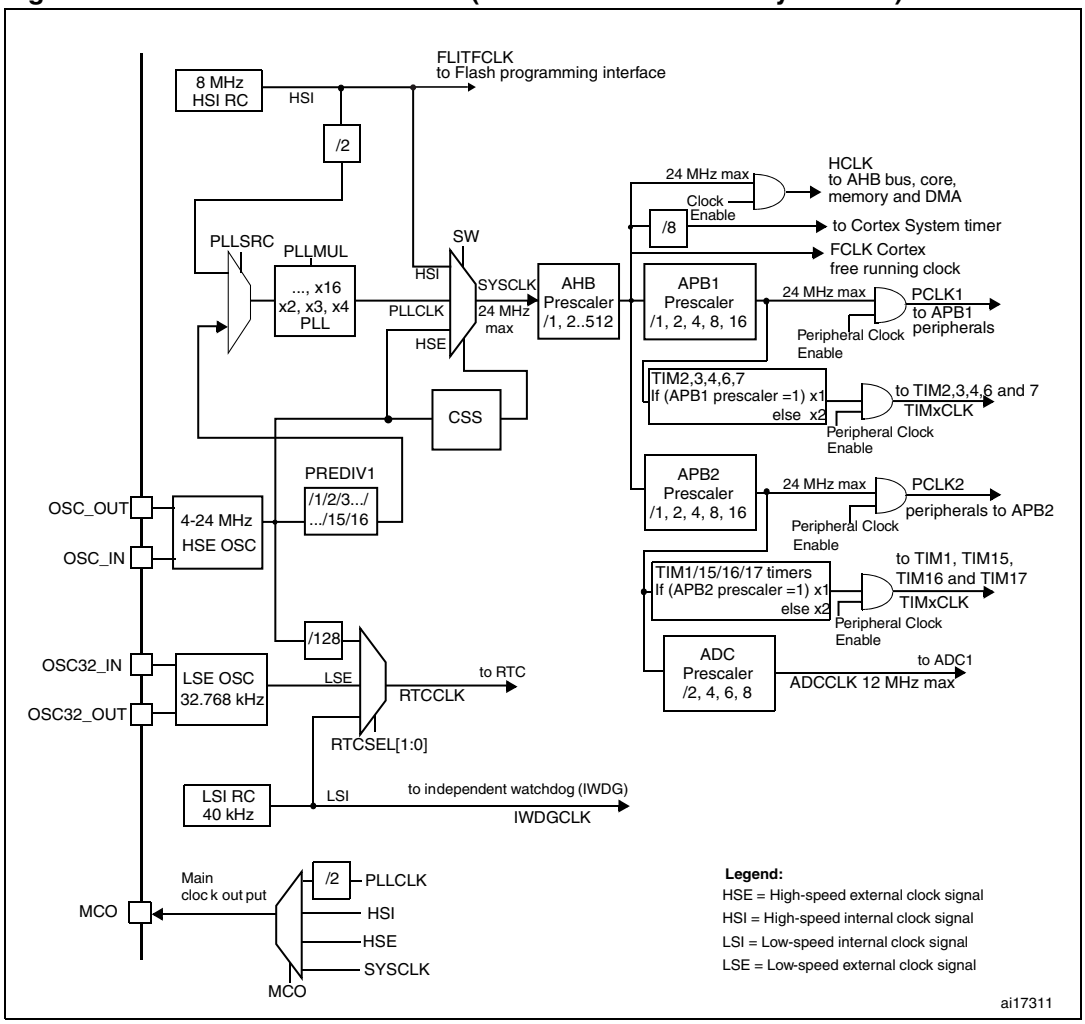
- HSI oscillator clock
- HSE oscillator clock
- PLL clock

The devices have the following two secondary clock sources:

- 40 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.
- 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the real-time clock (RTCCLK)

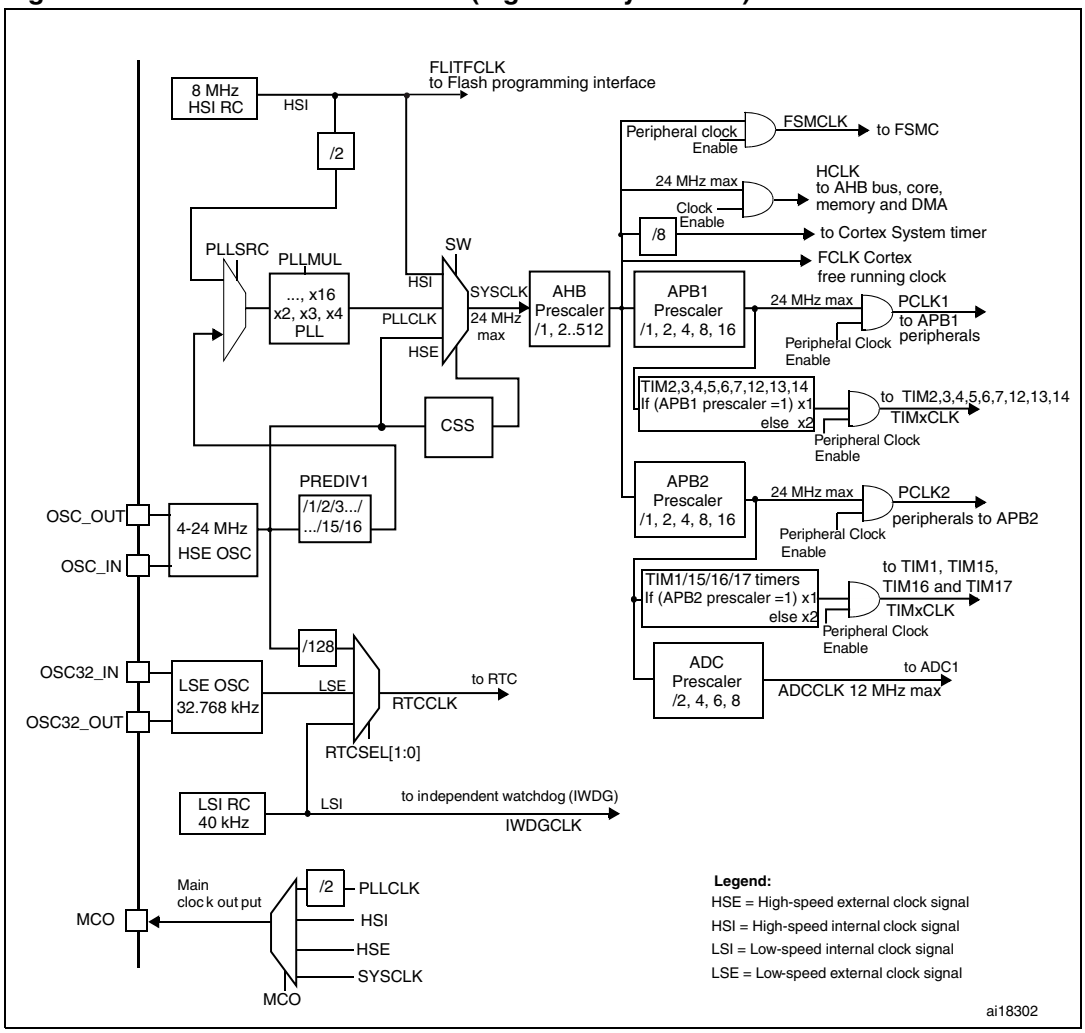
Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Figure 8. STM32F100xx clock tree (low and medium-density devices)



1. For full details about the internal and external clock source characteristics, please refer to the “Electrical characteristics” section in your device datasheet.

Figure 9. STM32F100xx clock tree (high-density devices)



Several prescalers allow the configuration of the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB, APB1 and APB2 domains is 24 MHz.

The RCC feeds the Cortex System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the Cortex clock (HCLK), configurable in the SysTick Control and Status Register. ADC1 is clocked by the clock of the High Speed domain (APB2) divided by 2, 4, 6 or 8.

The Flash memory programming interface clock (FLITFCLK) is always the HSI clock.

The timer clock frequencies are automatically fixed by hardware. There are two cases:

1. if the APB prescaler is 1, the timer clock frequencies are set to the same frequency as that of the APB domain to which the timers are connected.
2. otherwise, they are set to twice (×2) the frequency of the APB domain to which the timers are connected.

FCLK acts as Cortex™-M3 free running clock. For more details refer to the ARM Cortex™-M3 Technical Reference

Manual.http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337e/DDI0337E_cortex_m3_r1p1_trm.pdf

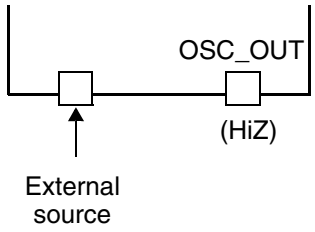
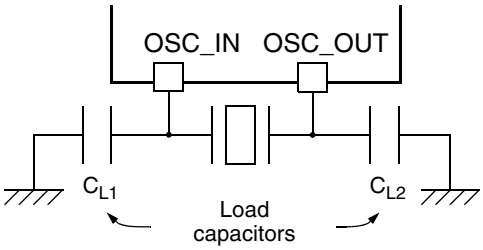
6.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

Figure 10. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 24 MHz. You select this mode by setting the HSEBYP and HSEON bits in the *Clock control register (RCC_CR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left hi-Z. See *Figure 10*.

External crystal/ceramic resonator (HSE crystal)

The 4 to 24 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in *Figure 10*. Refer to the electrical characteristics section of the *datasheet* for more details.

The HSERDY flag in the *Clock control register (RCC_CR)* indicates if the high-speed external oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.

The HSE Crystal can be switched on and off using the HSEON bit in the *Clock control register (RCC_CR)*.

6.2.2 HSI clock

The HSI clock signal is generated from an internal 8 MHz RC Oscillator and can be used directly as a system clock or divided by 2 to be used as PLL input.

The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

Calibration

RC oscillator frequencies 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 T_A=25°C.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *Clock control register (RCC_CR)*.

If the application is subject to voltage or temperature variations this may affect the RC oscillator speed. You can trim the HSI frequency in the application using the HSITRIM[4:0] bits in the *Clock control register (RCC_CR)*.

The HSIRDY flag in the *Clock control register (RCC_CR)* indicates if the HSI RC is stable or not. At startup, the HSI RC output clock is not released until this bit is set by hardware.

The HSI RC can be switched on and off using the HSION bit in the *Clock control register (RCC_CR)*.

The HSI signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 6.2.7: Clock security system (CSS) on page 75*.

6.2.3 PLL

The internal PLL can be used to multiply the HSI RC output or HSE oscillator divided by 1..16 output clock frequency. Refer to *Figure 8* and *Clock control register (RCC_CR)*.

The PLL configuration (selection of HSI oscillator divided by 2 or HSE oscillator for PLL input clock, and multiplication factor) must be done before enabling the PLL. Once the PLL enabled, these parameters cannot be changed.

Note: The PLL output frequency must be in the range of 16-24 MHz.

An interrupt can be generated when the PLL is ready if enabled in the *Clock interrupt register (RCC_CIR)*.

6.2.4 LSE clock

The LSE crystal is a 32.768 kHz Low Speed External crystal or ceramic resonator. It has the advantage providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in *Backup domain control register (RCC_BDCR)*.

The LSERDY flag in the *Backup domain control register (RCC_BDCR)* indicates if the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.

External source (LSE bypass)

In this mode, an external clock source must be provided. It must have a frequency of 32.768 kHz. You select this mode by setting the LSEBYP and LSEON bits in the *Backup domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left Hi-Z. See *Figure 10*.

6.2.5 LSI clock

The LSI RC acts as an low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and Auto-wakeup unit (AWU). The clock frequency is around 40 kHz. For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the *Control/status register (RCC_CSR)*.

The LSIRDY flag in the *Control/status register (RCC_CSR)* indicates if the low-speed internal oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC_CIR)*.

LSI calibration

Note: LSI calibration is only available on high-density value line devices.

The frequency dispersion of the Low Speed Internal RC (LSI) oscillator can be calibrated to

have accurate RTC time base and/or IWDG timeout (when LSI is used as clock source for these peripherals) with an acceptable accuracy.

This calibration is performed by measuring the LSI clock frequency with respect to TIM5 input clock (TIM5CLK). According to this measurement done at the precision of the HSE oscillator, the software can adjust the programmable 20-bit prescaler of the RTC to get an accurate time base or can compute accurate IWDG timeout.

Use the following procedure to calibrate the LSI:

1. Enable TIM5 timer and configure channel4 in input capture mode
2. Set the TIM5CH4_IREMAP bit in the AFIO_MAPR register to connect the LSI clock internally to TIM5 channel4 input capture for calibration purposes.
3. Measure the frequency of LSI clock using the TIM5 Capture/compare 4 event or interrupt.
4. Use the measured LSI frequency to update the 20-bit prescaler of the RTC depending on the desired time base and/or to compute the IWDG timeout.

6.2.6 System clock (SYSCLK) selection

After a system reset, the HSI oscillator is selected as system clock. When a clock source is used directly or through the PLL as 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 will be ready. Status bits in the *Clock control register (RCC_CR)* indicate which clock(s) is (are) ready and which clock is currently used as system clock.

6.2.7 Clock security system (CSS)

Clock Security System can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled, a clock failure event is sent to the break input of the advanced-control timers (TIM1) 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 Cortex™-M3 NMI (Non-Maskable Interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and an NMI is automatically generated. The NMI will be 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 Clock interrupt register (RCC_CIR).

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 HSI oscillator 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.

6.2.8 RTC clock

The RTCCLK clock source can be either the HSE/128, LSE or LSI clocks. This is selected by programming the RTCSEL[1:0] bits in the *Backup domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the Backup domain.

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 V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If LSI is selected as Auto-Wakeup unit (AWU) clock:
  - The AWU state is not guaranteed if the V_{DD} supply is powered off. Refer to *Section 6.2.5: LSI clock on page 74* for more details on LSI calibration.
- If the HSE clock divided by 128 is used as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.8 V domain).
  - The DPB bit (Disable backup domain write protection) in the Power controller register must be set to 1 (refer to *Section 4.4.1: Power control register (PWR_CR)*).

6.2.9 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

6.2.10 Clock-out capability

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. The configuration registers of the corresponding GPIO port must be programmed in alternate function mode. One of 4 clock signals can be selected as the MCO clock.

- SYSCLK
- HSI
- HSE
- PLL clock divided by 2

The selection is controlled by the MCO[2:0] bits of the *Clock configuration register (RCC_CFGR)*.

6.3 RCC registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

6.3.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Bits 31:26 Reserved, always read as 0.

Bit 25 **PLLRDY:** PLL clock ready flag

Set by hardware to indicate that the PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PLLON:** PLL enable

Set and cleared by software to enable PLL.

Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock.

0: PLL OFF

1: PLL ON

Bits 23:20 Reserved, always read as 0.

Bit 19 **CSSON:** Clock security system enable

Set and cleared by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if an HSE clock failure is detected.

0: Clock detector OFF

1: Clock detector ON (Clock detector ON if the HSE oscillator is ready , OFF if not).

Bit 18 **HSEBYP:** External high-speed clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit set, to be used by the device. The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE oscillator not bypassed

1: HSE oscillator oscillator bypassed with external clock

Bit 17 **HSERDY:** External high-speed clock ready flag

Set by hardware to indicate that the HSE oscillator is stable. This bit needs 6 cycles of the HSE oscillator clock to go to zero after HSEON is reset.

0: HSE oscillator not ready

1: HSE oscillator ready

Bit 16 **HSEON:** External high-speed clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering in Stop or Standby 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 15:8 **HSICAL[7:0]:** Internal high-speed clock calibration

These bits are initialized automatically at startup.

Bits 7:3 **HSITRIM[4:0]:** Internal high-speed 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 internal HSI RC.

The default value is 16, which, when added to the HSICAL value, should trim the HSI to 8 MHz ± 1%. The trimming step ($F_{hsitrim}$) is around 40 kHz between two consecutive HSICAL steps.

Bit 2 Reserved, always read as 0.

Bit 1 **HSIRDY:** Internal high-speed clock ready flag

Set by hardware to indicate that internal 8 MHz RC oscillator is stable. After the HSION bit is cleared, HSIRDY goes low after 6 internal 8 MHz RC oscillator clock cycles.

0: internal 8 MHz RC oscillator not ready

1: internal 8 MHz RC oscillator ready

Bit 0 **HSION:** Internal high-speed clock enable

Set and cleared by software.

Set by hardware to force the internal 8 MHz RC oscillator ON when leaving Stop or Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock. This bit cannot be reset if the internal 8 MHz RC is used directly or indirectly as system clock or is selected to become the system clock.

0: internal 8 MHz RC oscillator OFF

1: internal 8 MHz RC oscillator ON

6.3.2 Clock configuration register (RCC_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Reserved		PLLMUL[3:0]				PLL XTPRE	PLL SRC
					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
ADC PRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bits 31:27 Reserved, always read as 0.

Bits 26:24 **MCO:** Microcontroller clock output

Set and cleared by software.

0xx: No clock

100: System clock (SYSCLK) selected

101: HSI clock selected

110: HSE clock selected

111: PLL clock divided by 2 selected

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:22 Reserved.

Bits 21:18 **PLLMUL:** PLL multiplication factor

These bits are written by software to define the PLL multiplication factor. These bits can be written only when PLL is disabled.

Caution: The PLL output frequency must be in the 16-24 MHz range.

0000: PLL input clock x 2

0001: PLL input clock x 3

0010: PLL input clock x 4

0011: PLL input clock x 5

0100: PLL input clock x 6

0101: PLL input clock x 7

0110: PLL input clock x 8

0111: PLL input clock x 9

1000: PLL input clock x 10

1001: PLL input clock x 11

1010: PLL input clock x 12

1011: PLL input clock x 13

1100: PLL input clock x 14

1101: PLL input clock x 15

1110: PLL input clock x 16

1111: PLL input clock x 16

Bit 17 **PLLXTPRE:** LSB of division factor PREDIV1

Set and cleared by software to select the least significant bit of the PREDIV1 division factor. It is the same bit as bit 0 in the RCC_CFGR2 register, so modifying bit 0 in the RCC_CFGR2 register changes this bit accordingly.

If bits[3:1] in the RCC_CFGR2 register are not set, the PLLXTPRE bit controls if PREDIV1 divides its input clock by 2 (PLLXTPRE=1) or not (PLLXTPRE=0).

This bit can be written only when the PLL is disabled.

Bit 16 **PLLSRC:** PLL entry clock source

Set and cleared by software to select PLL clock source. This bit can be written only when PLL is disabled.

0: HSI oscillator clock / 2 selected as PLL input clock

1: Clock from PREDIV1 selected as the PLL input clock

Bits 15:14 **ADCPRE:** ADC prescaler

Set and cleared by software to select the frequency of the clock to ADC1.

00: PLCK2 divided by 2

01: PLCK2 divided by 4

10: PLCK2 divided by 6

11: PLCK2 divided by 8

Bits 13:11 **PPRE2:** APB high-speed prescaler (APB2)

Set and cleared by software to control the division factor of the APB high-speed clock (PCLK2).

0xx: HCLK not divided

100: HCLK divided by 2

101: HCLK divided by 4

110: HCLK divided by 8

111: HCLK divided by 16

Bits 10:8 **PPRE1:** APB low-speed prescaler (APB1)

Set and cleared by software to control the division factor of the APB low-speed clock (PCLK1).

Warning: the software has to set correctly these bits to not exceed 36 MHz on this domain.

0xx: HCLK not divided

100: HCLK divided by 2

101: HCLK divided by 4

110: HCLK divided by 8

111: HCLK divided by 16

Bits 7:4 **HPRE:** AHB prescaler

Set and cleared by software to control the division factor of the AHB clock.

0xxx: SYSCLK not divided

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

Bits 3:2 **SWS:** System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.
00: HSI oscillator used as system clock
01: HSE oscillator used as system clock
10: PLL used as system clock
11: not applicable

Bits 1:0 **SW:** System clock switch

Set and cleared by software to select SYSCLK source.
Set by hardware to force HSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).
00: HSI selected as system clock
01: HSE selected as system clock
10: PLL selected as system clock
11: not allowed

6.3.3 Clock interrupt register (RCC_CIR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	Reserved		PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Reserved		PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
			rw	rw	rw	rw	rw	r			r	r	r	r	r

Bits 31:24 Reserved, always read as 0.

Bit 23 **CSSC:** Clock security system interrupt clear

This bit is set by software to clear the CSSF flag.
0: No effect
1: Clear CSSF flag

Bits 22:21 Reserved, always read as 0.

Bit 20 **PLLRDYC:** PLL ready interrupt clear

This bit is set by software to clear the PLLRDYF flag.
0: No effect
1: PLLRDYF cleared

Bit 19 **HSERDYC:** HSE ready interrupt clear

This bit is set by software to clear the HSERDYF flag.
0: No effect
1: HSERDYF cleared

Bit 18 **HSIRDYC:** HSI ready interrupt clear
This bit is set software to clear the HSIRDYF flag.
0: No effect
1: HSIRDYF cleared

Bit 17 **LSERDYC:** LSE ready interrupt clear
This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF cleared

Bit 16 **LSIRDYC:** LSI ready interrupt clear
This bit is set by software to clear the LSIRDYF flag.
0: No effect
1: LSIRDYF cleared

Bits 15:13 Reserved, always read as 0.

Bit 12 **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 11 **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 10 **HSIRDYIE:** HSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the internal 8 MHz RC oscillator stabilization.
0: HSI ready interrupt disabled
1: HSI ready interrupt enabled

Bit 9 **LSERDYIE:** LSE ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the external 32 kHz oscillator stabilization.
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled

Bit 8 **LSIRDYIE:** LSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by internal RC 40 kHz oscillator stabilization.
0: LSI ready interrupt disabled
1: LSI ready interrupt enabled

Bit 7 **CSSF:** Clock security system interrupt flag
Set by hardware when a failure is detected in the HSE oscillator.
Cleared by software setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure

Bits 6:5 Reserved, always read as 0.

Bit 4 **PLLRDYF:** PLL ready interrupt flag
Set by hardware when the PLL locks and PLLRDYDIE is set.
Cleared by software setting the PLLRDYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock

Bit3 **HSERDYF:** HSE ready interrupt flag
Set by hardware when External Low Speed clock becomes stable and HSERDYDIE is set.
Cleared by software setting the HSERDYC bit.
0: No clock ready interrupt caused by the HSE oscillator
1: Clock ready interrupt caused by the HSE oscillator

Bit 2 **HSIRDYF:** HSI ready interrupt flag
Set by hardware when the Internal High Speed clock becomes stable and HSIRDYDIE is set.
Cleared by software setting the HSIRDYC bit.
0: No clock ready interrupt caused by the internal 8 MHz RC oscillator
1: Clock ready interrupt caused by the internal 8 MHz RC oscillator

Bit 1 **LSERDYF:** LSE ready interrupt flag
Set by hardware when the External Low Speed clock becomes stable and LSERDYDIE is set.
Cleared by software setting the LSERDYC bit.
0: No clock ready interrupt caused by the external 32 kHz oscillator
1: Clock ready interrupt caused by the external 32 kHz oscillator

Bit 0 **LSIRDYF:** LSI ready interrupt flag
Set by hardware when the internal low speed clock becomes stable and LSIRDYDIE is set.
Cleared by software setting the LSIRDYC bit.
0: No clock ready interrupt caused by the internal RC 40 kHz oscillator
1: Clock ready interrupt caused by the internal RC 40 kHz oscillator

6.3.4 APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x0C

Reset value: 0x00000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM17 RST	TIM16 RST	TIM15 RST
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	Res.	ADC1 RST	IOPG RST	IOPF RST	IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.	AFIO RST
	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw

Bits 31:19 Reserved, always read as 0.

Bit 18 **TIM17RST:** TIM17 reset
Set and cleared by software.
0: No effect
1: Reset TIM17

Bit 17 **TIM16RST:** TIM16 reset
Set and cleared by software.
0: No effect
1: Resets TIM16

Bit 16 **TIM15RST:** TIM15 reset
Set and cleared by software.
0: No effect
1: Resets TIM15

Bit 15 Reserved.

Bit 14 **USART1RST:** USART1 reset
Set and cleared by software.
0: No effect
1: Reset USART1

Bit 13 Reserved.

Bit 12 **SPI1RST:** SPI 1 reset
Set and cleared by software.
0: No effect
1: Reset SPI 1

Bit 11 **TIM1RST:** TIM1 timer reset
Set and cleared by software.
0: No effect
1: Reset TIM1 timer

Bit 10 Reserved.

Bit 9 **ADC1RST:** ADC 1 interface reset
Set and cleared by software.
0: No effect
1: Reset ADC 1 interface

Bit 8 **IOPGRST:** IO port G reset
Set and cleared by software.
0: No effect
1: Reset IO port G

Bit 7 **IOPFRST:** IO port F reset
Set and cleared by software.
0: No effect
1: Reset I/O port F

Bit 6 **IOPERST:** IO port E reset
Set and cleared by software.
0: No effect
1: Reset IO port E

Bit 5 **IOPDRST:** IO port D reset
Set and cleared by software.
0: No effect
1: Reset I/O port D

Bit 4 **IOPCRST:** IO port C reset
Set and cleared by software.
0: No effect
1: Reset I/O port C

Bit 3 **IOPBRST:** IO port B reset
Set and cleared by software.
0: No effect
1: Reset I/O port B

Bit 2 **IOPARST:** I/O port A reset
Set and cleared by software.
0: No effect
1: Reset I/O port A

Bit 1 Reserved, always read as 0.

Bit 0 **AFIORST:** Alternate function I/O reset
Set and cleared by software.
0: No effect
1: Reset Alternate Function

6.3.5 APB1 peripheral reset register (RCC_APB1RSTR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CECRST	DACRST	PWRRST	BKPRST	Reserved				I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.
	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
SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, always read as 0.

Bit 30 **CECRST:** CEC reset
Set and cleared by software.
0: No effect
1: Reset CEC

Bit 29 **DACRST:** DAC interface reset
Set and cleared by software.
0: No effect
1: Reset DAC interface

Bit 28 **PWRRST:** Power interface reset
Set and cleared by software.
0: No effect
1: Reset power interface

Bit 27 **BKPRST:** Backup interface reset
Set and cleared by software.
0: No effect
1: Reset backup interface

Bits 26:23 Reserved, always read as 0.

Bit 22 **I2C2RST:** I2C 2 reset
Set and cleared by software.
0: No effect
1: Reset I2C 2

Bit 21 **I2C1RST:** I2C 1 reset
Set and cleared by software.
0: No effect
1: Reset I2C 1

Bit 20 **UART5RST:** UART 5 reset
Set and cleared by software.
0: No effect
1: Reset UART 5

Bit 19 **UART4RST:** UART 4 reset
Set and cleared by software.
0: No effect
1: Reset UART 4

Bit 18 **USART3RST:** USART 3 reset
Set and cleared by software.
0: No effect
1: Reset USART 3

Bit 17 **USART2RST:** USART 2 reset
Set and cleared by software.
0: No effect
1: Reset USART 2

Bit 16 Reserved, always read as 0.

Bit 15 **SPI3RST:** SPI 3 reset
Set and cleared by software.
0: No effect
1: Reset SPI 3

Bit 14 **SPI2RST:** SPI 2 reset
Set and cleared by software.
0: No effect
1: Reset SPI 2

Bits 13:12 Reserved, always read as 0.

Bit 11 **WWDGRST:** Window watchdog reset
Set and cleared by software.
0: No effect
1: Reset window watchdog

Bits 10:9 Reserved, always read as 0.

Bit 8 **TIM14RST:** Timer 14 reset
Set and cleared by software.
0: No effect
1: Reset timer 14

Bit 7 **TIM13RST:** Timer 13 reset
Set and cleared by software.
0: No effect
1: Reset timer 13

Bit 6 **TIM12RST:** Timer 12 reset
Set and cleared by software.
0: No effect
1: Reset timer 12

Bit 5 **TIM7RST:** Timer 7 reset
Set and cleared by software.
0: No effect
1: Reset timer 7

Bit 4 **TIM6RST:** Timer 6 reset
Set and cleared by software.
0: No effect
1: Reset timer 6

Bit 3 **TIM5RST:** Timer 5 reset
Set and cleared by software.
0: No effect
1: Reset timer 5

Bit 2 **TIM4RST:** Timer 4 reset
Set and cleared by software.
0: No effect
1: Reset timer 4

Bit 1 **TIM3RST:** Timer 3 reset
Set and cleared by software.
0: No effect
1: Reset timer 3

Bit 0 **TIM2RST:** Timer 2 reset
Set and cleared by software.
0: No effect
1: Reset timer 2

6.3.6 AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x14

Reset value: 0x0000 0014

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							FSMCEN	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN	DMA2EN	DMA1EN
							rw		rw		rw		rw	rw	rw

Bits 31:9 Reserved, always read as 0.

Bit 8 **FSMCEN:** FSMC clock enable
Set and cleared by software.
0: FSMC clock disabled
1: FSMC clock enabled

Bit 7 Reserved, always read as 0.

Bit 6 **CRCEN:** CRC clock enable
Set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled

Bit 5 Reserved, always read as 0.

Bit 4 **FLITFEN:** FLITF clock enable
Set and cleared by software to disable/enable FLITF clock during sleep mode.
0: FLITF clock disabled during Sleep mode
1: FLITF clock enabled during Sleep mode

Bit 3 Reserved, always read as 0.

Bit 2 **SRAMEN:** SRAM interface clock enable
Set and cleared by software to disable/enable SRAM interface clock during Sleep mode.
0: SRAM interface clock disabled during Sleep mode.
1: SRAM interface clock enabled during Sleep mode

Bit 1 **DMA2EN:** DMA2 clock enable
Set and cleared by software.
0: DMA2 clock disabled
1: DMA2 clock enabled

Bit 0 **DMA1EN:** DMA1 clock enable
Set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled

6.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM17 EN	TIM16 EN	TIM15 EN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	Res.	SPI1 EN	TIM1 EN	Res.	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:19 Reserved, always read as 0.

Bit 18 **TIM17EN:** TIM17 Timer clock enable
Set and cleared by software.
0: TIM17 timer clock disabled
1: TIM17 timer clock enabled

Bit 17 **TIM16EN:** TIM16 timer clock enable
Set and cleared by software.
0: TIM16 timer clock disabled
1: TIM16 timer clock enabled

Bit 16 **TIM15EN:** TIM15 timer clock enable
Set and cleared by software.
0: TIM15 timer clock disabled
1: TIM15 timer clock enabled

Bit 15 Reserved.

Bit 14 **USART1EN:** USART1 clock enable
Set and cleared by software.
0: USART1 clock disabled
1: USART1 clock enabled

Bit 13 Reserved.

Bit 12 **SPI1EN:** SPI 1 clock enable
Set and cleared by software.
0: SPI 1 clock disabled
1: SPI 1 clock enabled

Bit 11 **TIM1EN:** TIM1 Timer clock enable
Set and cleared by software.
0: TIM1 timer clock disabled
1: TIM1 timer clock enabled

Bit 10 Reserved.

Bit 9 **ADC1EN:** ADC 1 interface clock enable
Set and cleared by software.
0: ADC 1 interface disabled
1: ADC 1 interface clock enabled

Bit 8 **IOPGEN:** I/O port G clock enable
Set and cleared by software.
0: I/O port G clock disabled
1: I/O port G clock enabled

Bit 7 **IOPFEN:** I/O port F clock enable
Set and cleared by software.
0: I/O port F clock disabled
1: I/O port F clock enabled

Bit 6 **IOPEEN:** I/O port E clock enable
Set and cleared by software.
0: I/O port E clock disabled
1: I/O port E clock enabled

Bit 5 **IOPDEN:** I/O port D clock enable
Set and cleared by software.
0: I/O port D clock disabled
1: I/O port D clock enabled

Bit 4 **IOPCEN:** I/O port C clock enable
Set and cleared by software.
0: I/O port C clock disabled
1:I/O port C clock enabled

Bit 3 **IOPBEN:** I/O port B clock enable
Set and cleared by software.
0: I/O port B clock disabled
1:I/O port B clock enabled

Bit 2 **IOPAEN:** I/O port A clock enable
Set and cleared by software.
0: I/O port A clock disabled
1:I/O port A clock enabled

Bit 1 Reserved, always read as 0.

Bit 0 **AFIOEN:** Alternate function I/O clock enable
Set and cleared by software.
0: Alternate Function I/O clock disabled
1:Alternate Function I/O clock enabled

6.3.8 APB1 peripheral clock enable register (RCC_APB1ENR)

Address: 0x1C

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait state, except if the access occurs while an access to a peripheral on APB1 domain is on going. In this case, wait states are inserted until this access to APB1 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CECEN	DACEN	PWREN	BKPEN	Reserved				I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, always read as 0.

Bit 30 **CECEN:** CEC clock enable
Set and cleared by software.
0: CEC clock disabled
1: CEC clock enabled

Bit 29 **DACEN:** DAC interface clock enable
Set and cleared by software.
0: DAC interface clock disabled
1: DAC interface clock enable

Bit 28 **PWREN:** Power interface clock enable
Set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enable

Bit 27 **BKPEN:** Backup interface clock enable
Set and cleared by software.
0: Backup interface clock disabled
1: Backup interface clock enabled

Bits 26:23 Reserved, always read as 0.

Bit 22 **I2C2EN:** I2C 2 clock enable
Set and cleared by software.
0: I2C 2 clock disabled
1: I2C 2 clock enabled

Bit 21 **I2C1EN:** I2C 1 clock enable
Set and cleared by software.
0: I2C 1 clock disabled
1: I2C 1 clock enabled

Bit 20 **UART5EN:** UART 5 clock enable
Set and cleared by software.
0: UART 5 clock disabled
1: UART 5 clock enabled

Bit 19 **UART4EN:** UART 4 clock enable
Set and cleared by software.
0: UART 4 clock disabled
1: UART 4 clock enabled

Bit 18 **USART3EN:** USART 3 clock enable
Set and cleared by software.
0: USART 3 clock disabled
1: USART 3 clock enabled

Bit 17 **USART2EN:** USART 2 clock enable
Set and cleared by software.
0: USART 2 clock disabled
1: USART 2 clock enabled

Bits 16 Reserved, always read as 0.

Bit 15 **SPI3EN:** SPI 3 clock enable
Set and cleared by software.
0: SPI 3 clock disabled
1: SPI 3 clock enabled

Bit 14 **SPI2EN:** SPI 2 clock enable
Set and cleared by software.
0: SPI 2 clock disabled
1: SPI 2 clock enabled

Bits 13:12 Reserved, always read as 0.

Bit 11 **WWDGEN:** Window watchdog clock enable
Set and cleared by software.
0: Window watchdog clock disabled
1: Window watchdog clock enabled

Bits 10:9 Reserved, always read as 0.

Bit 8 **TIM14EN:** Timer 14 clock enable
Set and cleared by software.
0: Timer 14 clock disabled
1: Timer 14 clock enabled

Bit 7 **TIM13EN:** Timer 13 clock enable
Set and cleared by software.
0: Timer 13 clock disabled
1: Timer 13 clock enabled

Bit 6 **TIM12EN:** Timer 12 clock enable
Set and cleared by software.
0: Timer 12 clock disabled
1: Timer 12 clock enabled

Bit 5 **TIM7EN:** Timer 7clock enable
Set and cleared by software.
0: Timer 7 clock disabled
1: Timer 7 clock enabled

Bit 4 **TIM6EN:** Timer 6 clock enable
Set and cleared by software.
0: Timer 6 clock disabled
1: Timer 6 clock enabled

Bit 3 **TIM5EN:** Timer 5 clock enable
Set and cleared by software.
0: Timer 5 clock disabled
1: Timer 5 clock enabled

Bit 2 **TIM4EN:** Timer 4 clock enable
Set and cleared by software.
0: Timer 4 clock disabled
1: Timer 4 clock enabled

Bit 1 **TIM3EN:** Timer 3 clock enable
Set and cleared by software.
0: Timer 3 clock disabled
1: Timer 3 clock enabled

Bit 0 **TIM2EN:** Timer 2 clock enable
Set and cleared by software.
0: Timer 2 clock disabled
1: Timer 2 clock enabled

6.3.9 Backup domain control register (RCC_BDCR)

Address offset: 0x20
Reset value: 0x0000 0000, reset by Backup domain 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: LSEON, LSEBYP, RTCSEL and RTCEN bits of the Backup domain control register (RCC_BDCR) are in the Backup domain. As a result, after Reset, these bits are write-protected and the DBP bit in the Power control register (PWR_CR) has to be set before these can be modified. Refer to Section 5: Backup registers (BKP) for further information. These bits are only reset after a Backup domain Reset (see Section 6.1.3: Backup domain reset). Any internal or external Reset will not have any effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Reserved					RTCSEL[1:0]		Reserved					LSE BYP	LSE RDY	LSEON
rw						rw	rw						rw	r	rw

Bits 31:17 Reserved, always read as 0.

Bit 16 **BDRST:** Backup domain software reset
Set and cleared by software.
0: Reset not activated
1: Resets the entire Backup domain

Bit 15 **RTCEN:** RTC clock enable
Set and cleared by software.
0: RTC clock disabled
1: RTC clock enabled

Bits 14:10 Reserved, always read as 0.

Bits 9:8 **RTCSEL[1:0]:** RTC clock source selection
Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. The BDRST bit can be used to reset them.
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 128 used as RTC clock

Bits 7:3 Reserved, always read as 0.

Bit 2 **LSEBYP:** External low-speed oscillator bypass
Set and cleared by software to bypass oscillator in debug mode. This bit can be written only when the external 32 kHz oscillator is disabled.
0: LSE oscillator not bypassed
1: LSE oscillator bypassed

Bit 1 **LSERDY:** External low-speed 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 6 external low-speed oscillator clock cycles.

0: External 32 kHz oscillator not ready

1: External 32 kHz oscillator ready

Bit 0 **LSEON:** External low-speed oscillator enable

Set and cleared by software.

0: External 32 kHz oscillator OFF

1: External 32 kHz oscillator ON

6.3.10 Control/status register (RCC_CSR)

Address: 0x24

Reset value: 0x0C00 0000, reset by system Reset, except reset flags by power Reset only.

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Res.	RMVF	Reserved							
rw	rw	rw	rw	rw	rw		rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSI RDY	LSION
														r	rw

Bit 31 **LPWRRSTF:** Low-power reset flag

Set by hardware when a Low-power management reset occurs.
Cleared by writing to the RMVF bit.

0: No Low-power management reset occurred

1: Low-power management reset occurred

For further information on Low-power management reset, refer to *Low-power management reset*.

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 watchdog reset flag

Set by hardware when an independent watchdog reset from V_{DD} domain occurs.
Cleared by writing to the RMVF bit.

0: No watchdog reset occurred

1: 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 **PORRSTF:** POR/PDR reset flag
Set by hardware when a POR/PDR reset occurs.
Cleared by writing to the RMVF bit.
0: No POR/PDR reset occurred
1: POR/PDR reset 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 Reserved, always read as 0.

Bit 24 **RMVF:** Remove reset flag
Set by software to clear the reset flags.
0: No effect
1: Clear the reset flags

Bits 23:2 Reserved, always read as 0.

Bit 1 **LSIRDY:** Internal low-speed oscillator ready
Set and cleared by hardware to indicate when the internal RC 40 kHz oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 internal RC 40 kHz oscillator clock cycles.
0: Internal RC 40 kHz oscillator not ready
1: Internal RC 40 kHz oscillator ready

Bit 0 **LSION:** Internal low-speed oscillator enable
Set and cleared by software.
0: Internal RC 40 kHz oscillator OFF
1: Internal RC 40 kHz oscillator ON

6.3.11 Clock configuration register2 (RCC_CFGR2)

Address offset: 0x2C

Access: no wait state, word, half-word and byte access

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PREDIV1[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved.

Bits 3:0 **PREDIV1[3:0]:** PREDIV1 division factor

Set and cleared by software to select the PREDIV1 division factor. These bits can be written only when the PLL is disabled.

Note: Bit 0 is the same as bit 17 in the RCC_CFGR register, so modifying bit 17 in the RCC_CFGR register changes bit 0 accordingly.

0000: PREDIV1 input clock not divided
0001: PREDIV1 input clock divided by 2
0010: PREDIV1 input clock divided by 3
0011: PREDIV1 input clock divided by 4
0100: PREDIV1 input clock divided by 5
0101: PREDIV1 input clock divided by 6
0110: PREDIV1 input clock divided by 7
0111: PREDIV1 input clock divided by 8
1000: PREDIV1 input clock divided by 9
1001: PREDIV1 input clock divided by 10
1010: PREDIV1 input clock divided by 11
1011: PREDIV1 input clock divided by 12
1100: PREDIV1 input clock divided by 13
1101: PREDIV1 input clock divided by 14
1110: PREDIV1 input clock divided by 15
1111: PREDIV1 input clock divided by 16

6.3.12 RCC register map

The following table gives the RCC register map and the reset values.

Table 15. RCC 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	**RCC_CR**	Reserved						PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]								HSITRIM[4:0]					Reserved	HSIRDY	HSION
	Reset value							0	0					0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0		1	1
0x004	**RCC_CFGR**	Reserved					MCO [2:0]			Reserved	PLLMUL[3:0]					PLLXTPRE	PLLSRC	ADC PRE [1:0]		PPRE2 [2:0]			PPRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [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
0x008	**RCC_CIR**	Reserved								CSSC	Reserved		PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved			PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved		PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value									0			0	0	0	0	0				0	0	0	0	0	0			0	0	0	0	0
0x00C	**RCC_APB2RSTR**	Reserved													TIM17RST	TIM16RST	TIM15RST	Reserved		USART1RST	Reserved	SPI1RST	TIM1RST	Reserved	ADC1	IOPGRST	IOPFRST	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Reserved	AFIORST
	Reset value														0	0	0			0		0	0		0	0	0	0	0	0	0	0		0
0x010	**RCC_APB1RSTR**	Reserved	CECRST	DACRST	PWRRST	BKPRST	Reserved			I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Reserved		SPI3RST	SPI2RST	Reserved		WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value			0	0	0				0	0	0	0	0	0			0	0			0			0	0	0	0	0	0	0	0	0
0x014	**RCC_AHBENR**	Reserved																							FSMCEN	Reserved	CRCEN	Reserved	FLITFEN	Reserved	SRAMEN	DMA2EN	DMA1EN
	Reset value																								0		0		1		1	0	0
0x018	**RCC_APB2ENR**	Reserved													TIM17EN	TIM16EN	TIM15EN	Reserved		USART1EN	Reserved	SPI1EN	TIM1EN	Reserved	ADC1EN	IOPGEN	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Reserved	AFIOEN
	Reset value														0	0	0			0		0	0		0	0	0	0	0	0	0	0		0
0x01C	**RCC_APB1ENR**	Reserved	CECEN	DACEN	PWREN	BKPEN	Reserved			I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved		SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value		0	0	0	0				0	0	0	0	0	0			0	0			0			0	0	0	0	0	0	0	0	0
0x020	**RCC_BDCR**	Reserved															BDRST	RTCEN	Reserved					RTC SEL [1:0]		Reserved						LSEBYP	LSERDY	LSEON
	Reset value																0	0						0	0							0	0	0
0x024	**RCC_CSR**	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Reserved	RMVF	Reserved																							LSIRDY	LSION
	Reset value	0	0	0	0	1	1		0																							0	0
0x02C	**RCC_CFGR2**	Reserved																												PREDIV1[3:0]			
	Reset value																													0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

7 General-purpose and alternate-function I/Os (GPIOs and AFIOs)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

7.1 GPIO functional description

Each of the general-purpose I/O ports has two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH), two 32-bit data registers (GPIOx_IDR, GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 16-bit reset register (GPIOx_BRR) and a 32-bit locking register (GPIOx_LCKR).

Subject to the specific hardware characteristics of each I/O port listed in the *datasheet*, each port bit of the General Purpose IO (GPIO) Ports, can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words (half-word or byte accesses are not allowed). The purpose of the GPIOx_BSRR and GPIOx_BRR registers is to allow atomic read/modify accesses to any of the GPIO registers. This way, there is no risk that an IRQ occurs between the read and the modify access.

Figure 11 shows the basic structure of an I/O Port bit.

Figure 11. Basic structure of a standard I/O port bit

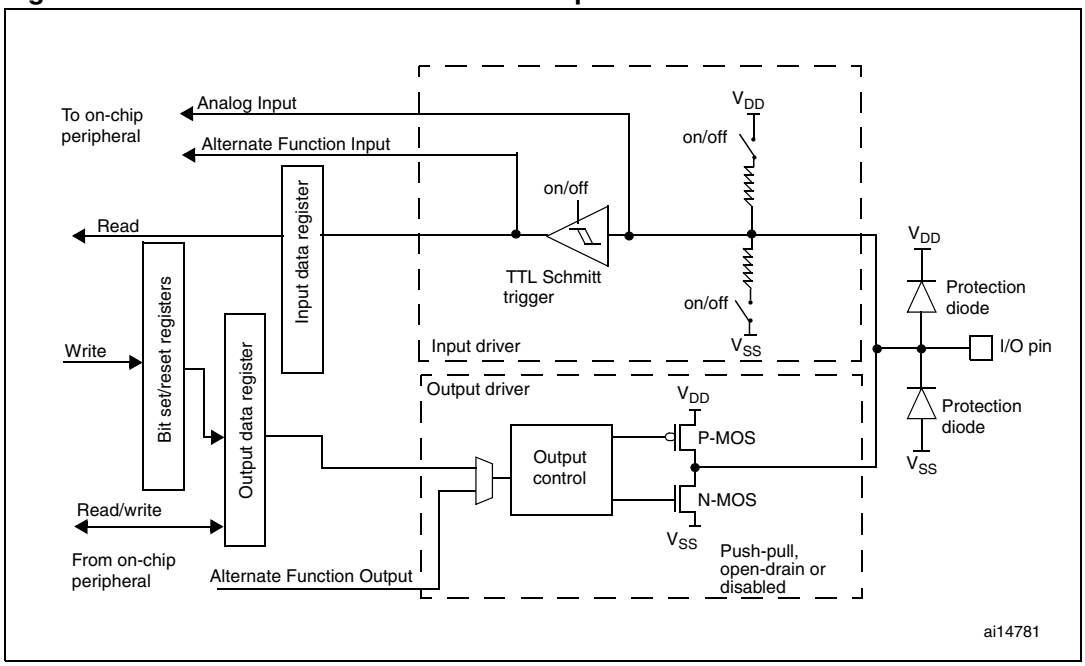
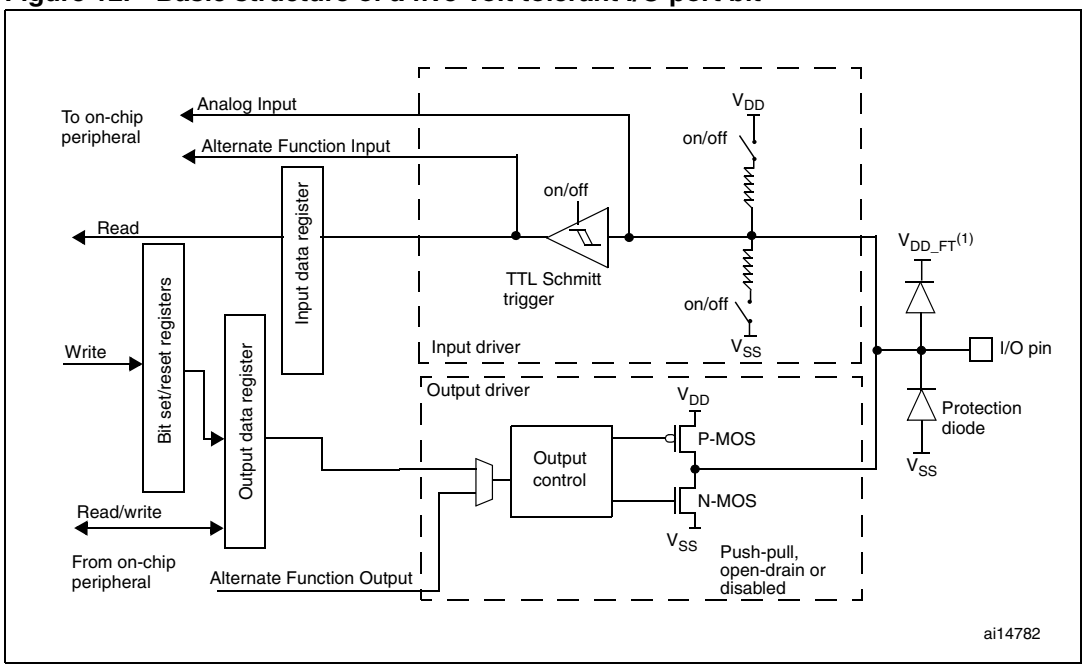


Figure 12. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 16. Port bit configuration table

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11 see *Table 17*		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			don't care
	Open-drain		1			don't care
Input	Analog	0	0	00		don't care
	Input floating		1			don't care
	Input pull-down	1	0			0
	Input pull-up					1

Table 17. Output MODE bits

MODE[1:0]	Meaning
00	Reserved
01	Max. output speed 10 MHz
10	Max. output speed 2 MHz
11	Max. output speed 50 MHz

7.1.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and the I/O ports are configured in Input Floating mode (CNFx[1:0]=01b, MODEx[1:0]=00b).

The JTAG pins are in input PU/PD after reset:

PA15: JTDI in PU

PA14: JTCK in PD

PA13: JTMS in PU

PB4: NJTRST in PU

When 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 N-MOS is activated when outputting 0).

The Input Data register (GPIOx_IDR) captures the data present on the I/O pin at every APB2 clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down which can be activated or not when configured as input.

7.1.2 Atomic bit set or reset

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify only one or several bits in a single atomic APB2 write access. This is achieved by programming to '1' the Bit Set/Reset Register (GPIOx_BSRR,

or for reset only GPIOx_BRR) to select the bits you want to modify. The unselected bits will not be modified.

7.1.3 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. For more information on external interrupts, refer to:

- *Section 8.2: External interrupt/event controller (EXTI) on page 132* and
- *Section 8.2.3: Wakeup event management on page 133*

7.1.4 Alternate functions (AF)

It is necessary to program the Port Bit Configuration Register before using a default alternate function.

- For alternate function inputs, the port must be configured in Input mode (floating, pull-up or pull-down) and the input pin must be driven externally.

Note: It is also possible to emulate the AFI input pin by software by programming the GPIO controller. In this case, the port should be configured in Alternate Function Output mode. And obviously, the corresponding port should not be driven externally as it will be driven by the software using the GPIO controller.

- For alternate function outputs, the port must be configured in Alternate Function Output mode (Push-Pull or Open-Drain).
- For bidirectional Alternate Functions, the port bit must be configured in Alternate Function Output mode (Push-Pull or Open-Drain). In this case the input driver is configured in input floating mode

If you configure a port bit as Alternate Function Output, this disconnects the output register and connects the pin to the output signal of an on-chip peripheral.

If software configures a GPIO pin as Alternate Function Output, but peripheral is not activated, its output is not specified.

7.1.5 Software remapping of I/O alternate functions

To optimize the number of peripheral I/O functions for different device packages, it is possible to remap some alternate functions to some other pins. This is achieved by software, by programming the corresponding registers (refer to *AFIO registers on page 119*. In that case, the alternate functions are no longer mapped to their original assignations.

7.1.6 GPIO locking mechanism

The locking mechanism allows the IO configuration to be frozen. When the LOCK sequence has been applied on a port bit, it is no longer possible to modify the value of the port bit until the next reset.

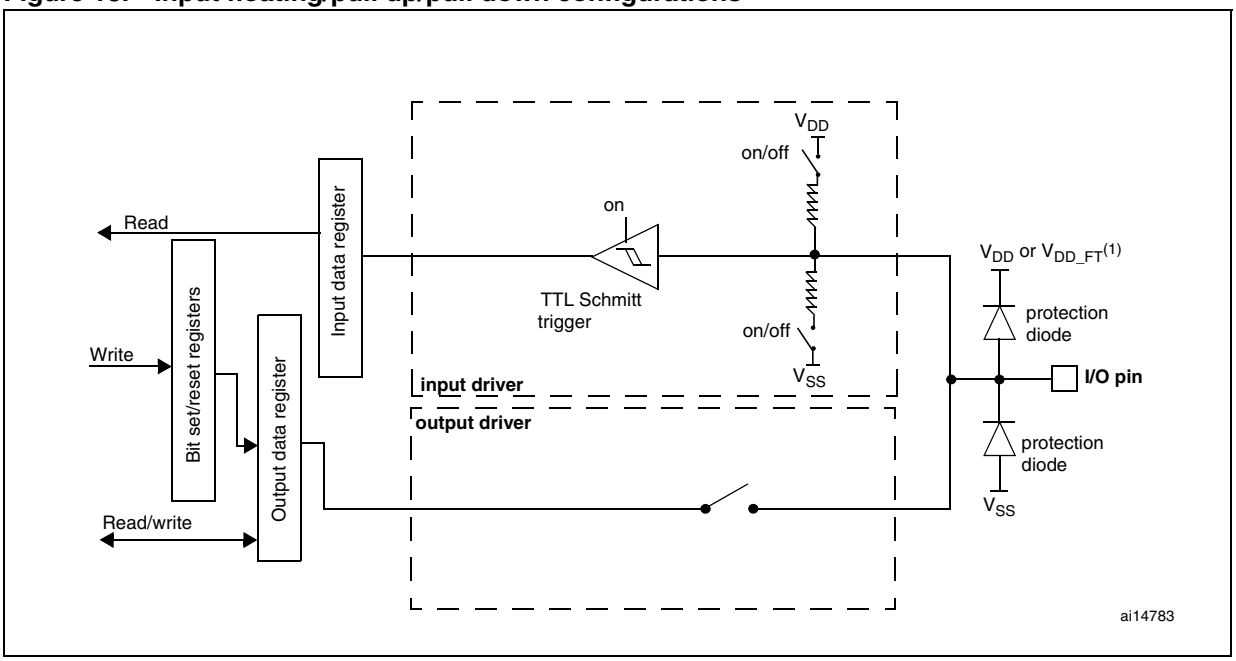
7.1.7 Input configuration

When the I/O Port is programmed as Input:

- The Output Buffer is disabled
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are activated or not depending on input configuration (pull-up, pull-down or floating):
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register obtains the I/O State.

The *Figure 13 on page 103* shows the Input Configuration of the I/O Port bit.

Figure 13. Input floating/pull up/pull down configurations



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

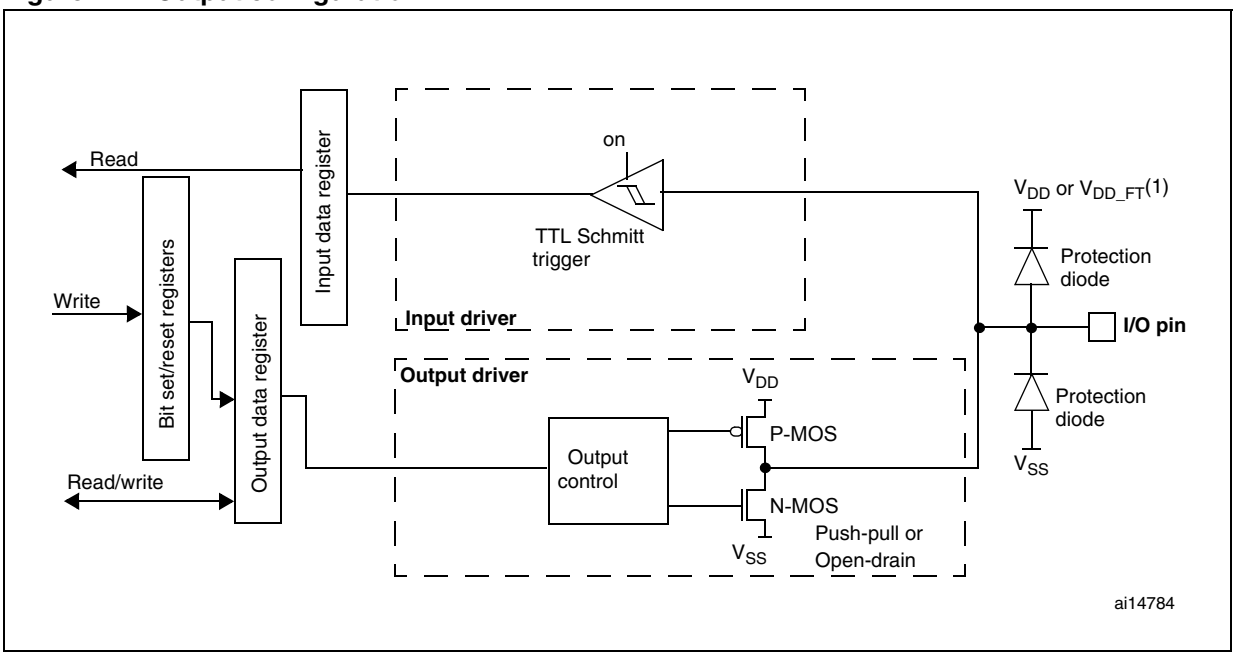
7.1.8 Output configuration

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 while 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 while a “1” in the Output register activates the P-MOS.
- The Schmitt Trigger Input is activated.
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in Push-Pull mode

The *Figure 14 on page 104* shows the Output configuration of the I/O Port bit.

Figure 14. Output configuration



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

7.1.9 Alternate function configuration

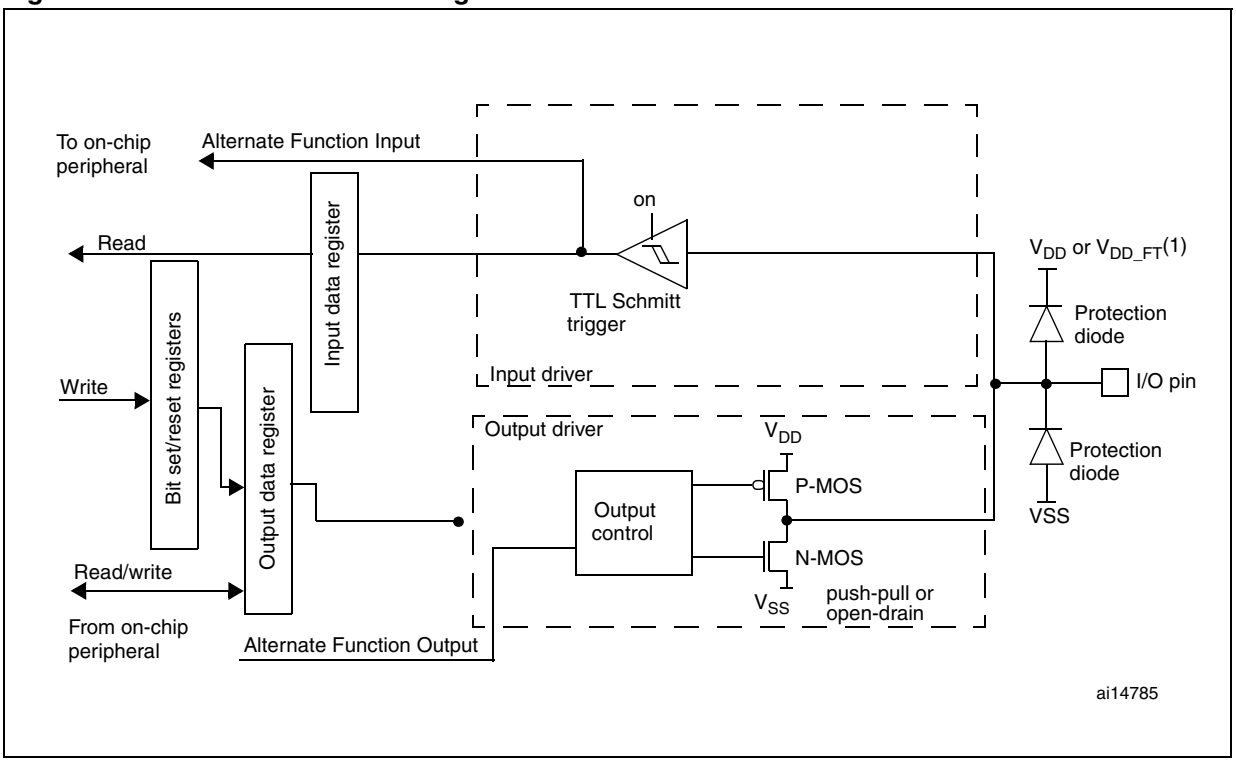
When the I/O Port is programmed as Alternate Function:

- The Output Buffer is turned on in Open Drain or Push-Pull configuration
- The Output Buffer is driven by the signal coming from the peripheral (alternate function out)
- The Schmitt Trigger Input is activated
- The weak pull-up and pull-down resistors are disabled.
- The data present on the I/O pin is sampled into the Input Data Register every APB2 clock cycle
- A read access to the Input Data Register gets the I/O state in open drain mode
- A read access to the Output Data register gets the last written value in Push-Pull mode

The *Figure 15 on page 105* shows the Alternate Function Configuration of the I/O Port bit. Also, refer to *Section 7.4: AFIO registers on page 119* for further information.

A set of Alternate Function I/O registers allow you to remap some alternate functions to different pins. Refer to

Figure 15. Alternate function configuration



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

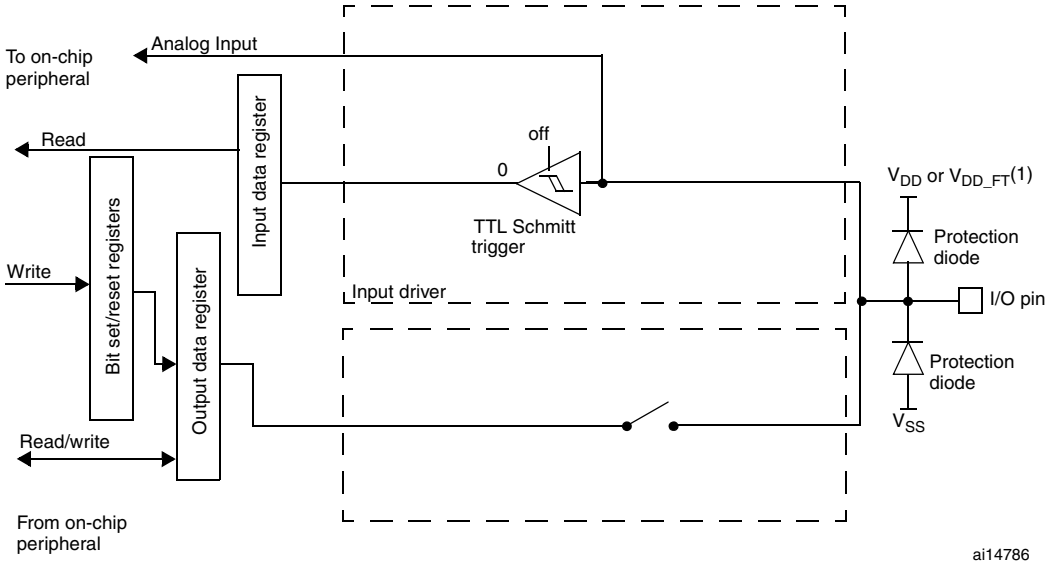
7.1.10 Analog configuration

When the I/O Port is programmed as Analog configuration:

- The Output Buffer is disabled.
- The Schmitt Trigger Input is de-activated 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.
- Read access to the Input Data Register gets the value “0”.

The *Figure 16 on page 106* shows the high impedance-analog configuration of the I/O Port bit.

Figure 16. High impedance-analog configuration



7.1.11 GPIO configurations for device peripherals

Table 18 to *Table 27* give the GPIO configurations of the device peripherals.

Table 18. Advanced timer TIM1

TIM1 pinout	Configuration	GPIO configuration
TIM1_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TIM1_CHxN	Complementary output channel x	Alternate function push-pull
TIM1_BKIN	Break input	Input floating
TIM1_ETR	External trigger timer input	Input floating

Table 19. General-purpose timers TIM2/3/4/5

TIM2/3/4/5 pinout	Configuration	GPIO configuration
TIM2/3/4/5_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TIM2/3/4/5_ETR	External trigger timer input	Input floating

Table 20. General-purpose timers TIM15/16/17

TIM15/16/17 pinout	Configuration	GPIO configuration
TIM15/16/17_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull

Table 20. General-purpose timers TIM15/16/17

TIM15/16/17 pinout	Configuration	GPIO configuration
TIM15/16/17_CHxN	Complementary output channel x	Alternate function push-pull
TIM15/16/17_BKIN	Break input	Input floating
TIM15/16/17_ETR	External trigger timer input	Input floating

Table 21. General-purpose timers TIM12/13/14

TIM12/13/14 pinout	Configuration	GPIO configuration
TIM12/13/14_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull

Table 22. USARTs

USART pinout	Configuration	GPIO configuration
USARTx_TX(1)	Full duplex	Alternate function push-pull
	Half duplex synchronous mode	Alternate function push-pull
USARTx_RX	Full duplex	Input floating / Input pull-up
	Half duplex synchronous mode	Not used. Can be used as a general IO
USARTx_CK	Synchronous mode	Alternate function push-pull
USARTx_RTS	Hardware flow control	Alternate function push-pull
USARTx_CTS	Hardware flow control	Input floating/ Input pull-up

1. The USART_TX pin can also beconfigured as alternate function open drain.

Table 23. SPI

SPI pinout	Configuration	GPIO configuration
SPIx_SCK	Master	Alternate function push-pull
	Slave	Input floating
SPIx_MOSI	Full duplex / master	Alternate function push-pull
	Full duplex / slave	Input floating / Input pull-up
	Simplex bidirectional data wire / master	Alternate function push-pull
	Simplex bidirectional data wire/ slave	Not used. Can be used as a GPIO

Table 23. SPI (continued)

SPI pinout	Configuration	GPIO configuration
SPIx_MISO	Full duplex / master	Input floating / Input pull-up
	Full duplex / slave (point to point)	Alternate function push-pull
	Full duplex / slave (multi-slave)	Alternate function open drain
	Simplex bidirectional data wire / master	Not used. Can be used as a GPIO
	Simplex bidirectional data wire/ slave (point to point)	Alternate function push-pull
	Simplex bidirectional data wire/ slave (multi-slave)	Alternate function open drain
SPIx_NSS	Hardware master /slave	Input floating/ Input pull-up / Input pull-down
	Hardware master/ NSS output enabled	Alternate function push-pull
	Software	Not used. Can be used as a GPIO

Table 24. CEC

CEC pinout	Configuration	GPIO configuration
CEC	CEC line	Alternate function open drain

Table 25. I2C

I2C pinout	Configuration	GPIO configuration
I2Cx_SCL	I2C clock	Alternate function open drain
I2Cx_SDA	I2C Data I/O	Alternate function open drain

The GPIO configuration of the ADC inputs should be analog.

Figure 17. ADC / DAC

ADC/DAC pin	GPIO configuration
ADC/DAC	Analog

Table 26. FSMC

FSMC pinout	GPIO configuration
FSMC_A[25:0] FSMC_D[15:0]	Alternate function push-pull
FSMC_CK	Alternate function push-pull
FSMC_NOE FSMC_NWE	Alternate function push-pull

Table 26. FSMC

FSMC pinout	GPIO configuration
FSMC_NE[4:1]	Alternate function push-pull
FSMC_NWAIT	Input floating/ Input pull-up
FSMC_NL FSMC_NBL[1:0]	Alternate function push-pull

Table 27. Other IOs

Pins	Alternate function	GPIO configuration
TAMPER-RTC pin	RTC output	Forced by hardware when configuring the BKP_CR and BKP_RTCCR registers
	Tamper event input	
MCO	Clock output	Alternate function push-pull
EXTI input lines	External input interrupts	Input floating / input pull-up / input pull-down

7.2 GPIO registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

7.2.1 Port configuration register low (GPIOx_CRL) (x=A..G)

Address offset: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
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
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 16: Port bit configuration table on page 101*.
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 16: Port bit configuration table on page 101*.
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

7.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
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
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 8 .. 15)

These bits are written by software to configure the corresponding I/O port.

Refer to *Table 16: Port bit configuration table on page 101*.

In input mode (MODE[1:0]=00):

00: Analog mode

01: Floating input (reset state)

10: Input with pull-up / pull-down

11: Reserved

In output mode (MODE[1:0] > 00):

00: General purpose output push-pull

01: General purpose output Open-drain

10: Alternate function output Push-pull

11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 8 .. 15)

These bits are written by software to configure the corresponding I/O port.

Refer to *Table 16: Port bit configuration table on page 101*.

00: Input mode (reset state)

01: Output mode, max speed 10 MHz.

10: Output mode, max speed 2 MHz.

11: Output mode, max speed 50 MHz.

7.2.3 Port input data register (GPIOx_IDR) (x=A..G)

Address offset: 0x08h

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **IDRy[15:0]:** Port input data (y= 0 .. 15)

These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.

7.2.4 Port output data register (GPIOx_ODR) (x=A..G)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
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 **ODRy[15:0]:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. E).

7.2.5 Port bit set/reset register (GPIOx_BSRR) (x=A..G)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x Reset *bit y (y= 0 .. 15)*

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x *Set bit y (y= 0 .. 15)*

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Set the corresponding ODRx bit

7.2.6 Port bit reset register (GPIOx_BRR) (x=A..G)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved

Bits 15:0 **BRy:** Port x Reset bit y (y= 0 .. 15)

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

7.2.7 Port configuration lock register (GPIOx_LCKR) (x=A..G)

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 it is no longer possible to modify the value of the port bit until the next reset.

Each lock bit freezes the corresponding 4 bits of the control register (CRL, CRH).

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved

Bit 16 **LCKK[16]:** Lock key

This bit can be read anytime. It can only be modified using the Lock Key Writing Sequence.
0: Port configuration lock key not active
1: Port configuration lock key active. GPIOx_LCKR register is locked until an MCU reset occurs.

LOCK key writing sequence:
Write 1
Write 0
Write 1
Read 0
Read 1 (this read is optional but confirms that the lock is active)

Note: During the LOCK Key Writing sequence, the value of LCK[15:0] must not change.
Any error in the lock sequence will abort the lock.

Bits 15:0 **LCKy:** Port x Lock bit y (y= 0 .. 15)

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.

7.3 Alternate function I/O and debug configuration (AFIO)

To optimize the number of peripherals available for the 64-pin or the 100-pin or the 144-pin package, it is possible to remap some alternate functions to some other pins. This is achieved by software, by programming the *AF remap and debug I/O configuration register (AFIO_MAPR) on page 121*. In this case, the alternate functions are no longer mapped to their original assignations.

7.3.1 Using OSC32_IN/OSC32_OUT pins as GPIO ports PC14/PC15

The LSE oscillator pins OSC32_IN and OSC32_OUT can be used as general-purpose I/O PC14 and PC15, respectively, when the LSE oscillator is off. The LSE has priority over the GP IOs function.

Note: 1 The PC14/PC15 GPIO functionality is lost when the 1.8 V domain is powered off (by entering standby mode) or when the backup domain is supplied by V_{BAT} (V_{DD} no more supplied). In this case the IOs are set in analog mode.

2 Refer to the note on IO usage restrictions in Section 4.1.2 on page 50.

7.3.2 Using OSC_IN/OSC_OUT pins as GPIO ports PD0/PD1

The HSE oscillator pins OSC_IN/OSC_OUT can be used as general-purpose I/O PD0/PD1 by programming the PD01_REMAP bit in the *AF remap and debug I/O configuration register (AFIO_MAPR)*.

This remap is available only on 48- and 64-pin packages (PD0 and PD1 are available on 100-pin and 144-pin packages, no need for remapping).

Note: The external interrupt/event function is not remapped. PD0 and PD1 cannot be used for external interrupt/event generation on 48- and 64-pin packages.

7.3.3 JTAG/SWD alternate function remapping

The debug interface signals are mapped on the GPIO ports as shown in *Table 28*.

Table 28. Debug interface signals

Alternate function	GPIO port
JTMS / SWDIO	PA13
JTCK / SWCLK	PA14
JTDI	PA15
JTDO / TRACESWO	PB3
NJTRST	PB4
TRACECK	PE2
TRACED0	PE3
TRACED1	PE4
TRACED2	PE5
TRACED3	PE6

To optimize the number of free GPIOs during debugging, this mapping can be configured in different ways by programming the SWJ_CFG[1:0] bits in the *AF remap and debug I/O configuration register (AFIO_MAPR)*. Refer to *Table 29*

Table 29. Debug port mapping

SWJ_CFG [2:0]	Available debug ports	SWJ I/O pin assigned				
		PA13 / JTMS/ SWDIO	PA14 / JTCK/S WCLK	PA15 / JTDI	PB3 / JTDO/ TRACE SWO	PB4/ NJTRST
000	Full SWJ (JTAG-DP + SW-DP) (Reset state)	X	X	X	X	X
001	Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	x	free
010	JTAG-DP Disabled and SW-DP Enabled	X	X	free	free(1)	free
100	JTAG-DP Disabled and SW-DP Disabled	free	free	free	free	free
Other	Forbidden					

1. Released only if not using asynchronous trace.

7.3.4 Timer alternate function remapping

Timer 4 channels 1 to 4 can be remapped from Port B to Port D. Other timer remapping possibilities are listed in *Table 35* to *Table 37*. Refer to *AF remap and debug I/O configuration register (AFIO_MAPR)*.

Table 30. TIM5 alternate function remapping(1)

Alternate function	TIM5CH4_IREMAP = 0	TIM5CH4_IREMAP = 1
TIM5_CH4	TIM5 Channel 4 is connected to PA3	LSI internal clock is connected to TIM5_CH4 input for calibration purpose.

1. Remap available only for high-density value line devices.

Table 31. TIM12 remapping(1)

Alternate function	TIM12_REMAP = 0	TIM12_REMAP = 1
TIM12_CH1	PC4	PB12
TIM12_CH2	PC5	PB13

1. Refer to the AF remap and debug I/O configuration register *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*. Remap available only for high-density value line devices.

Table 32. TIM13 remapping(1)

Alternate function	TIM13_REMAP = 0	TIM13_REMAP = 1
TIM13_CH1	PC8	PB0

1. Refer to the AF remap and debug I/O configuration register *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*. Remap available only for high-density value line devices.

Table 33. TIM14 remapping(1)

Alternate function	TIM14_REMAP = 0	TIM14_REMAP = 1
TIM14_CH1	PC9	PB1

1. Refer to the AF remap and debug I/O configuration register *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*. Remap available only for high-density value line devices.

Table 34. TIM4 alternate function remapping

Alternate function	TIM4_REMAP = 0	TIM4_REMAP = 1(1)
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD14
TIM4_CH4	PB9	PD15

1. Remap available only for 100-pin and for 144-pin package.

Table 35. TIM3 alternate function remapping

Alternate function	TIM3_REMAP[1:0] = “00” (no remap)	TIM3_REMAP[1:0] = “10” (partial remap)	TIM3_REMAP[1:0] = “11” (full remap) (1)
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

1. Remap available only for 64-pin, 100-pin and 144-pin packages.

Table 36. TIM2 alternate function remapping

Alternate function	TIM2_REMAP[1:0] = “00” (no remap)	TIM2_REMAP[1:0] = “01” (partial remap)	TIM2_REMAP[1:0] = “10” (partial remap)	TIM2_REMAP[1:0] = “11” (full remap)
TIM2_CH1_ETR(1)	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

1. TIM_CH1 and TIM_ETR share the same pin but cannot be used at the same time (which is why we have this notation: TIM2_CH1_ETR).

Table 37. TIM1 alternate function remapping

Alternate functions mapping	TIM1_REMAP[1:0] = “00” (no remap)	TIM1_REMAP[1:0] = “01” (partial remap)	TIM1_REMAP[1:0] = “11” (full remap) (1)
TIM1_ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13
TIM1_CH4	PA11		PE14
TIM1_BKIN	PB12	PA6	PE15
TIM1_CH1N	PB13	PA7	PE8
TIM1_CH2N	PB14 (2)	PB0	PE10
TIM1_CH3N	PB15 (2)	PB1	PE12

1. Remap available only for 100-pin and 144-pin packages.

Table 38. TIM1 DMA remapping(1)

DMA requests	TIM1_DMA_REMAP = 0	TIM1_DMA_REMAP = 1
TIM1_CH1 DMA request	Mapped on DMA1 Channel2	Mapped on DMA1 Channel6
TIM1_CH2 DMA request	Mapped on DMA1 Channel3	Mapped on DMA1 Channel6

1. Refer to the AF remap and debug I/O configuration register *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*.

Table 39. TIM15 remapping(1)

Alternate function	TIM15_REMAP = 0	TIM15_REMAP = 1
TIM15_CH1	PA2	PB14
TIM15_CH2	PA3	PB15

1. Refer to the AF remap and debug I/O configuration register *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*.

Table 40. TIM16 remapping(1)

Alternate function	TIM16_REMAP = 0	TIM16_REMAP = 1
TIM16_CH1	PB8	PA6

1. Refer to the AF remap and debug I/O configuration register *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*.

Table 41. TIM17 remapping(1)

Alternate function	TIM17_REMAP = 0	TIM17_REMAP = 1
TIM17_CH1	PB9	PA7

1. Refer to the AF remap and debug I/O configuration register *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*.

7.3.5 USART alternate function remapping

Refer to *AF remap and debug I/O configuration register (AFIO_MAPR)*.

Table 42. USART3 remapping

Alternate function	USART3_REMAP[1:0] = "00" (no remap)	USART3_REMAP[1:0] = "01" (partial remap) (1)	USART3_REMAP[1:0] = "11" (full remap) (2)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

1. Remap available only for 64-pin, 100-pin and 144-pin packages
2. Remap available only for 100-pin and 144-pin packages.

Table 43. USART2 remapping

Alternate functions	USART2_REMAP = 0	USART2_REMAP = 1(1)
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

1. Remap available only for 100-pin and 144-pin packages.

Table 44. USART1 remapping

Alternate function	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

7.3.6 I2C1 alternate function remapping

Refer to *AF remap and debug I/O configuration register (AFIO_MAPR)*

Table 45. I2C1 remapping

Alternate function	I2C1_REMAP = 0	I2C1_REMAP = 1
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

7.3.7 SPI1 alternate function remapping

Refer to *AF remap and debug I/O configuration register (AFIO_MAPR)*

Table 46. SPI1 remapping

Alternate function	SPI1_REMAP = 0	SPI1_REMAP = 1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

7.3.8 CEC remap

Refer to *Section 7.4.7: AF remap and debug I/O configuration register (AFIO_MAPR2)*.

Table 47. CEC remapping

Alternate function	CEC_REMAP = 0	CEC_REMAP = 1
CEC	PB8	PB10

7.4 AFIO registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

Note: To read/write the AFIO_EVCR, AFIO_MAPR, AFIO_MAPR2 and AFIO_EXTICRX registers, the AFIO clock should first be enabled. Refer to APB2 peripheral clock enable register (RCC_APB2ENR).

The peripheral registers have to be accessed by words (32-bit).

7.4.1 Event control register (AFIO_EVCR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EVOE	PORT[2:0]			PIN[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved

Bit 7 **EVOE:** Event output enable

Set and cleared by software. When set the EVENTOUT Cortex output is connected to the I/O selected by the PORT[2:0] and PIN[3:0] bits.

Bits 3:0 **PIN[3:0]:** Pin selection (x = A .. E)

Set and cleared by software. Select the pin used to output the Cortex EVENTOUT signal.

0000: Px0 selected
0001: Px1 selected
0010: Px2 selected
0011: Px3 selected
...
1111: Px15 selected

7.4.2 AF remap and debug I/O configuration register (AFIO_MAPR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					SWJ_CFG[2:0]			Reserved							TIM5CH4_IREMAP
					w	w	w								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	Reserved		TIM4_REMAP	TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved

Bits 26:24 **SWJ_CFG[2:0]:** Serial wire JTAG configuration

These bits are write-only (when read, the value is undefined). They are used to configure the SWJ and trace alternate function I/Os. The SWJ (Serial Wire JTAG) supports JTAG or SWD access to the Cortex debug port. The default state after reset is SWJ ON without trace. This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS / JTCK pin.

000: Full SWJ (JTAG-DP + SW-DP): Reset State
001: Full SWJ (JTAG-DP + SW-DP) but without NJTRST
010: JTAG-DP Disabled and SW-DP Enabled
100: JTAG-DP Disabled and SW-DP Disabled
Other combinations: no effect

Bits 23:17 Reserved.

Bit 15 **PD01_REMAP:** Port D0/Port D1 mapping on OSC_IN/OSC_OUT

This bit is set and cleared by software. It controls the mapping of PD0 and PD1 GPIO functionality. When the HSE oscillator is not used (application running on internal 8 MHz RC) PD0 and PD1 can be mapped on OSC_IN and OSC_OUT. This is available only on 48- and 64-pin packages (PD0 and PD1 are available on 100-pin packages, no need for remapping).

0: No remapping of PD0 and PD1
1: PD0 remapped on OSC_IN, PD1 remapped on OSC_OUT,

Bits 14:13 Reserved.

Bit 12 **TIM4_REMAP:** TIM4 remapping

This bit is set and cleared by software. It controls the mapping of TIM4 channels 1 to 4 onto the GPIO ports.

0: No remap (TIM4_CH1/PB6, TIM4_CH2/PB7, TIM4_CH3/PB8, TIM4_CH4/PB9)
1: Full remap (TIM4_CH1/PD12, TIM4_CH2/PD13, TIM4_CH3/PD14, TIM4_CH4/PD15)

Note: TIM4_ETR on PE0 is not re-mapped.

Bits 11:10 **TIM3_REMAP[1:0]:** TIM3 remapping

These bits are set and cleared by software. They control the mapping of TIM3 channels 1 to 4 on the GPIO ports.

00: No remap (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1)
01: Not used
10: Partial remap (CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1)
11: Full remap (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9)

Note: TIM3_ETR on PE0 is not re-mapped.

Bits 9:8 **TIM2_REMAP[1:0]:** TIM2 remapping

These bits are set and cleared by software. They control the mapping of TIM2 channels 1 to 4 and external trigger (ETR) on the GPIO ports.
00: No remap (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3)
01: Partial remap (CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3)
10: Partial remap (CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11)
11: Full remap (CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)

Bits 7:6 **TIM1_REMAP[1:0]:** TIM1 remapping

These bits are set and cleared by software. They control the mapping of TIM1 channels 1 to 4, 1N to 3N, external trigger (ETR) and Break input (BKIN) on the GPIO ports.
00: No remap (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15)
01: Partial remap (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1)
10: not used
11: Full remap (ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)

Bits 5:4 **USART3_REMAP[1:0]:** USART3 remapping

These bits are set and cleared by software. They control the mapping of USART3 CTS, RTS,CK,TX and RX alternate functions on the GPIO ports.
00: No remap (TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14)
01: Partial remap (TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14)
10: not used
11: Full remap (TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)

Bit 3 **USART2_REMAP:** USART2 remapping

This bit is set and cleared by software. It controls the mapping of USART2 CTS, RTS,CK,TX and RX alternate functions on the GPIO ports.
0: No remap (CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4)
1: Remap (CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)

Bit 2 **USART1_REMAP:** USART1 remapping

This bit is set and cleared by software. It controls the mapping of USART1 TX and RX alternate functions on the GPIO ports.
0: No remap (TX/PA9, RX/PA10)
1: Remap (TX/PB6, RX/PB7)

Bit 1 **I2C1_REMAP:** I2C1 remapping

This bit is set and cleared by software. It controls the mapping of I2C1 SCL and SDA alternate functions on the GPIO ports.
0: No remap (SCL/PB6, SDA/PB7)
1: Remap (SCL/PB8, SDA/PB9)

Bit 0 **SPI1_REMAP:** SPI1 remapping

This bit is set and cleared by software. It controls the mapping of SPI1 NSS, SCK, MISO, MOSI alternate functions on the GPIO ports.
0: No remap (NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7)
1: Remap (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5)

7.4.3 External interrupt configuration register 1 (AFIO_EXTICR1)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]:** EXTI x configuration (x= 0 to 3)

These bits are written by software to select the source input for EXTIx external interrupt.
Refer to *Section 8.2.5: External interrupt/event line mapping on page 134*
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

7.4.4 External interrupt configuration register 2 (AFIO_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]:** EXTI x configuration (x= 4 to 7)

These bits are written by software to select the source input for EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

7.4.5 External interrupt configuration register 3 (AFIO_EXTICR3)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x= 8 to 11)

These bits are written by software to select the source input for EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

7.4.6 External interrupt configuration register 4 (AFIO_EXTICR4)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x= 12 to 15)

These bits are written by software to select the source input for EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin

7.4.7 AF remap and debug I/O configuration register (AFIO_MAPR2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		MISC_ REMAP	TIM12_ REMAP	TIM67_ DAC_ DMA_ REMAP	FSMC_ NADV	TIM14_ REMAP	TIM13_ REMAP	Reserved			TIM1_ DMA_ REMAP	CEC_ REMAP	TIM17_ REMAP	TIM16_ REMAP	TIM15_ REMAP
		rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:14 Reserved.

Bit 13 **MISC_REMAP:** Miscellaneous features remapping.

This bit is set and cleared by software. It controls miscellaneous features

The DMA2 channel 5 interrupt position in the vector table

The timer selection for DAC trigger 3 (TSEL[2:0] = 011, for more details refer to the DAC_CR register).

0: DMA2 channel 5 interrupt is mapped with DMA2 channel 4 at position 59, TIM5 TRGO event is selected as DAC Trigger 3, TIM5 triggers TIM1/3.

1: DMA2 channel 5 interrupt is mapped separately at position 60 and TIM15 TRGO event is selected as DAC Trigger 3, TIM15 triggers TIM1/3.

Note: This bit is available only in high density value line devices.

Bit 12 **TIM12_REMAP:** TIM12 remapping

This bit is set and cleared by software. It controls the mapping of the TIM12_CH1 and TIM12_CH2 alternate function onto the GPIO ports.

0: No remap (CH1/PC4, CH2/PC5)

1: Remap (CH1/PB12, CH2/PB13)

Note: This bit is available only in high density value line devices.

Bit 11 **TIM76_DAC_DMA_REMAP:** TIM67_DAC DMA remapping

This bit is set and cleared by software. It controls the mapping of the TIM6_DAC1 and TIM7_DAC2 DMA requests onto the DMA1 channels.

0: No remap (TIM6_DAC1 DMA request/DMA2 Channel3, TIM7_DAC2 DMA request/DMA2 Channel4)

1: Remap (TIM6_DAC1 DMA request/DMA1 Channel3, TIM7_DAC2 DMA request/DMA1 Channel4)

Bit 10 **FSMC_NADV:** NADV connect/disconnect

This bit is set and cleared by software. It controls the use of the optional FSMC_NADV signal.

0: The NADV signal is connected to the output (default)

1: The NADV signal is *not* connected. The I/O pin can be used by another peripheral.

Note: This bit is available only in high density value line devices.

Bit 9 **TIM14_REMAP:** TIM14 remapping

This bit is set and cleared by software. It controls the mapping of the TIM14_CH1 alternate function onto the GPIO ports.

0: No remap (PC9)

1: Remap (PB1)

Bit 8 **TIM13_REMAP:** TIM13 remapping

This bit is set and cleared by software. It controls the mapping of the TIM13_CH1 alternate function onto the GPIO ports.

0: No remap (PC8)

1: Remap (PB0)

Bits 7:5 Reserved.

Bit 4 **TIM1_DMA_REMAP:** TIM1 DMA remapping

This bit is set and cleared by software. It controls the mapping of the TIM1 channel 1 and channel 2 DMA requests onto the DMA1 channels.

0: No remap (TIM1_CH1 DMA request/DMA1 Channel2, TIM1_CH2 DMA request/DMA1 Channel3)

1: Remap (TIM1_CH1 DMA request/DMA1 Channel6, TIM1_CH2 DMA request/DMA1 Channel6)

Bit 3 **CEC_REMAP:** CEC remapping

This bit is set and cleared by software. It controls the mapping of the alternate functions of the CEC line onto the GPIO ports.

0: No remap (CEC/PB8)

1: Remap (CEC/PB10)

Bit 2 **TIM17_REMAP:** TIM17 remapping

This bit is set and cleared by software. It controls the mapping of the alternate functions of TIM17 channel 1 onto the GPIO ports.

0: No remap (CH1/PB9)

1: Remap (CH1/PA7)

Bit 1 **TIM16_REMAP:** TIM16 remapping

This bit is set and cleared by software. It controls the mapping of the alternate functions of TIM16 channel 1 onto the GPIO ports.

0: No remap (CH1/PB8)

1: Remap (CH1/PA6)

Bit 0 **TIM15_REMAP:** TIM15 remapping

This bit is set and cleared by software. It controls the mapping of the alternate functions of TIM15 channels 1 and 2 onto the GPIO ports.

0: No remap (CH1/PA2, CH2/PA3)

1: Remap (CH1/PB14, CH2/PB15)

7.5 GPIO and AFIO register maps

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses. The following tables give the GPIO and AFIO register map and the reset values.

Table 48. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOx_CRL**	CNF7 [1:0]		MODE7 [1:0]		CNF6 [1:0]		MODE6 [1:0]		CNF5 [1:0]		MODE5 [1:0]		CNF4 [1:0]		MODE4 [1:0]		CNF3 [1:0]		MODE3 [1:0]		CNF2 [1:0]		MODE2 [1:0]		CNF1 [1:0]		MODE1 [1:0]		CNF0 [1:0]		MODE0 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x04	**GPIOx_CRH**	CNF 15 [1:0]		MODE15 [1:0]		CNF 14 [1:0]		MODE14 [1:0]		CNF 13 [1:0]		MODE13 [1:0]		CNF 12 [1:0]		MODE12 [1:0]		CNF 11 [1:0]		MODE11 [1:0]		CNF 10 [1:0]		MODE10 [1:0]		CNF 9 [1:0]		MODE9 [1:0]		CNF 8 [1:0]		MODE8 [1:0]	
	Reset value	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0x08	**GPIOx_IDR**	Reserved																IDR[15:0]I															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOx_ODR**	Reserved																ODR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_BSRR**	BR[15:0]																BSR[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	0
0x14	**GPIOx_BRR**	Reserved																BR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_LCKR**	Reserved															LCKK	LCK[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49. AFIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**AFIO_EVCR**	Reserved																									EVOE	PORT[2:0]			PIN[3:0]		
	Reset value																										0	0	0	0	0	0	0
0x04	**AFIO_MAPR**	Reserved				SWJ_CFG[2:0]			Reserved								TIM5CH4_IREMAP	PD01_REMAP	Reserved		TIM4_REMAP		TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP
	Reset value					0	0	0									0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**AFIO_EXTICR1**	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**AFIO_EXTICR2**	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**AFIO_EXTICR3**	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**AFIO_EXTICR4**	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49. AFIO 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
0x1C	**AFIO_MAPR2**	Reserved																		MISC_ REMAP	TIM12_REMAP	TIM67_DAC_ DMA_ REMAP	FSMC_NADV	TIM14_REMAP	TIM13_REMAP	Reserved			TIM1_DMA_REMAP	CEC_REMAP	TIM17_REMAP	TIM16_REMAP	TIM15_REMAP
	Reset value																			0	0	0	0	0	0				0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

8 Interrupts and events

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This Section applies to the whole STM32F100xx family, unless otherwise specified.

8.1 Nested vectored interrupt controller (NVIC)

Features

- 60 maskable interrupt channels in high-density value line devices and 56 in low and medium-density value line devices (not including the sixteen Cortex™-M3 interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to *STM32F100xx Cortex-M3 programming manual* (see Related documents on page 1).

8.1.1 SysTick calibration value register

The SysTick calibration value is set to 9000, which gives a reference time base of 3 ms with the SysTick clock set to 3 MHz (max HCLK/8).

8.1.2 Interrupt and exception vectors

Table 50. Vector table for STM32F100xx devices

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI_Handler	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault_Handler	All class of fault	0x0000_000C

Table 50. Vector table for STM32F100xx devices (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	0	settable	MemManage_Handler	Memory management	0x0000_0010
	1	settable	BusFault_Handler	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault_Handler	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C - 0x0000_002B
	3	settable	SVC_Handler	System service call via SWI instruction	0x0000_002C
	4	settable	DebugMon_Handler	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV_Handler	Pendable request for system service	0x0000_0038
	6	settable	SysTick_Handler	System tick timer	0x0000_003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER_STAMP	Tamper and TimeStamp through EXTI line interrupts	0x0000_0048
3	10	settable	RTC_WKUP	RTC Wakeup through EXTI line interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1	ADC1 global interrupt	0x0000_0088
-	-	-	-	Reserved	0x0000_008C - 0x0000_0098

Table 50. Vector table for STM32F100xx devices (continued)

Position	Priority	Type of priority	Acronym	Description	Address
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	TIM1_BRK_TIM15	TIM1 Break and TIM15 global interrupt	0x0000_00A0
25	32	settable	TIM1_UP_TIM16	TIM1 Update and TIM16 global interrupts	0x0000_00A4
26	33	settable	TIM1_TRG_COM_TIM17	TIM1 Trigger and Commutation and TIM17 global interrupts	0x0000_00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I^2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I^2C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I^2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I^2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000_00E4
42	49	settable	CEC	CEC global interrupt	0x0000_00E8
43	50	settable	TIM12	TIM12 global interrupt	0x0000_00EC
44	51	settable	TIM13	TIM13 global interrupt	0x0000_00F0
45	52	settable	TIM14	TIM14 global interrupt	0x0000_00F4
-	-	-	-	Reserved	0x0000_00F8 - 0x0000_00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000_0100
-	-	-	-	Reserved	0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114

Table 50. Vector table for STM32F100xx devices (continued)

Position	Priority	Type of priority	Acronym	Description	Address
54	61	settable	TIM6_DAC	TIM6 global and DAC underrun interrupts	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4_5	DMA2 Channel4 and DMA2 Channel5 global interrupts	0x0000_012C
60	67	settable	DMA2_Channel5(1)	DMA2 Channel5 global interrupt	0x0000_0130

1. For High-density value line devices, the DMA2 Channel 5 is mapped at postion 60 only if the MISC_REMAP bit in the AFIO_MAPR2 register is set and DMA2 Channel 2 is connected with DMA2 Channel 4 at position 59 when the MISC_REMAP bit in the AFIO_MAPR2 register is reset.

8.2 External interrupt/event controller (EXTI)

The external interrupt/event controller consists of up to 18 edge detectors for generating event/interrupt requests. Each input line can be independently configured to select the type (pulse or pending) and the corresponding trigger event (rising or falling or both). Each line can also masked independently. A pending register maintains the status line of the interrupt requests

8.2.1 Main features

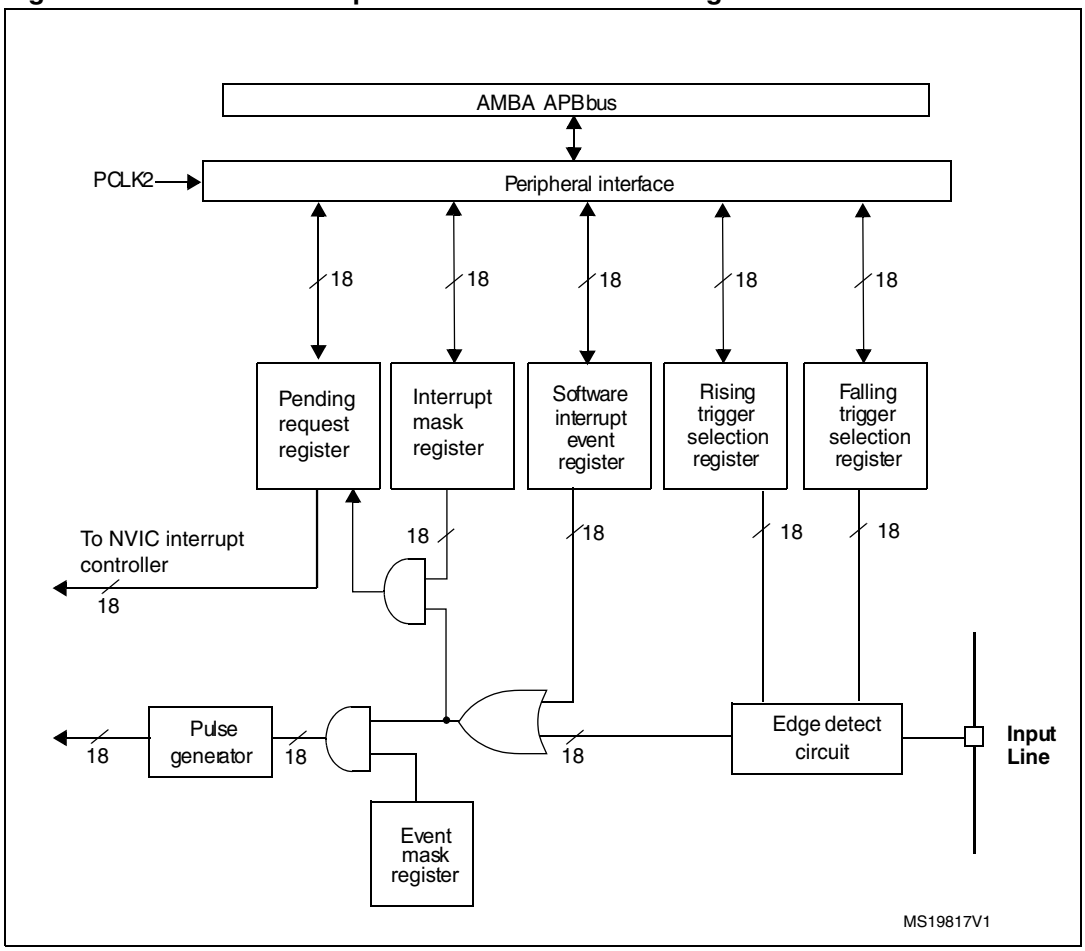
The EXTI controller main features are the following:

- Independent trigger and mask on each interrupt/event line
- Dedicated status bit for each interrupt line
- Generation of up to 18 software event/interrupt requests
- Detection of external signal with pulse width lower than APB2 clock period. Refer to the electrical characteristics section of the datasheet for details on this parameter.

8.2.2 Block diagram

The block diagram is shown in *Figure 18*.

Figure 18. External interrupt/event controller block diagram



8.2.3 Wakeup event management

The STM32F100xx is able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated either by:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex-M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

To use an external line as a wakeup event, refer to *Section 8.2.4: Functional description*.

8.2.4 Functional description

To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1’ to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is

generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1’ in the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1’ to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set

An interrupt/event request can also be generated by software by writing a ‘1’ in the software interrupt/event register.

Hardware interrupt selection

To configure the 18 lines as interrupt sources, use the following procedure:

- Configure the mask bits of the 18 Interrupt lines (EXTI_IMR)
- Configure the Trigger Selection bits of the Interrupt lines (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the External Interrupt Controller (EXTI) so that an interrupt coming from one of the 18 lines can be correctly acknowledged.

Hardware event selection

To configure the 18 lines as event sources, use the following procedure:

- Configure the mask bits of the 18 Event lines (EXTI_EMR)
- Configure the Trigger Selection bits of the Event lines (EXTI_RTSR and EXTI_FTSR)

Software interrupt/event selection

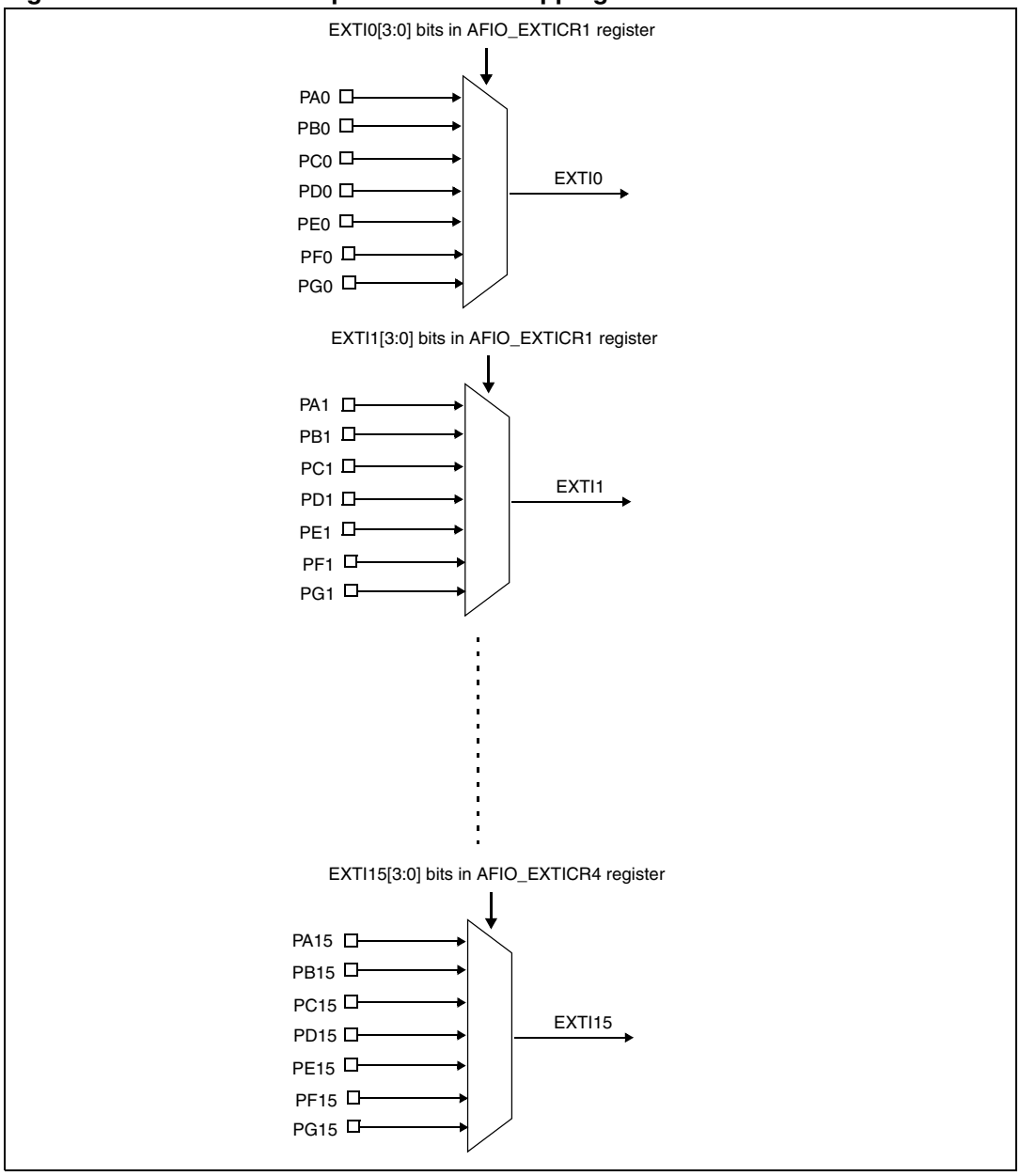
The 18 lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

- Configure the mask bits of the 18 Interrupt/Event lines (EXTI_IMR, EXTI_EMR)
- Set the required bit of the software interrupt register (EXTI_SWIER)

8.2.5 External interrupt/event line mapping

The 112 GPIOs are connected to the 16 external interrupt/event lines in the following manner:

Figure 19. External interrupt/event GPIO mapping



1. To configure the AFIO_EXTICRx for the mapping of external interrupt/event lines onto GPIOs, the AFIO clock should first be enabled. Refer to *Section 6.3.7: APB2 peripheral clock enable register (RCC_APB2ENR)*.

The two other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event

8.3 EXTI registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

8.3.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														MR17	MR16
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value (0).

Bits 17:0 **MRx:** Interrupt Mask on line x

0: Interrupt request from Line x is masked
1: Interrupt request from Line x is not masked

8.3.2 Event mask register (EXTI_EMR)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														MR17	MR16
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value (0).

Bits 17:0 **MRx:** Event mask on line x

0: Event request from Line x is masked
1: Event request from Line x is not masked

8.3.3 Rising trigger selection register (EXTI_RTSR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														TR17	TR16
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value (0).

Bits 17:0 **TRx:** Rising trigger event configuration bit of line x
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

Note: The external wakeup lines are edge triggered, no glitches must be generated on these lines. If a rising edge on external interrupt line occurs during writing of EXTI_RTSR register, the pending bit will not be set.

Rising and Falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

8.3.4 Falling trigger selection register (EXTI_FTSR)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														TR17	TR16
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value (0).

Bits 17:0 **TRx:** Falling trigger event configuration bit of line x
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line

Note: The external wakeup lines are edge triggered, no glitches must be generated on these lines. If a falling edge on external interrupt line occurs during writing of EXTI_FTSR register, the pending bit will not be set.

Rising and Falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

8.3.5 Software interrupt event register (EXTI_SWIER)

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														SWIER 17	SWIER 16
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value (0).

Bits 17:0 **SWIERx:** Software interrupt on line x

Writing a 1 to this bit when it is at 0 sets the corresponding pending bit in EXTI_PR. If the interrupt is enabled on this line on the EXTI_IMR and EXTI_EMR, an interrupt request is generated.

This bit is cleared by clearing the corresponding bit of EXTI_PR (by writing a 1 into the bit)

8.3.6 Pending register (EXTI_PR)

Address offset: 0x14
Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														PR17	PR16
														rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
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

Bits 31:18 Reserved, must be kept at reset value (0).

Bits 17:0 **PRx:** Pending bit

0: No trigger request occurred
1: selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

8.3.7 EXTI register map

The following table gives the EXTI register map and the reset values.

Table 51. External interrupt/event controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**EXTI_IMR**	Reserved														MR[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**EXTI_EMR**	Reserved														EMR[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**EXTI_RTSR**	Reserved														RTSR[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**EXTI_FTSR**	Reserved														FTSR[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**EXTI_SWIER**	Reserved														SWIER[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_PR**	Reserved														PR[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

9 Direct memory access controller (DMA)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

9.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The two DMA controllers have 12 channels in total (7 for DMA1 and 5 for DMA2), each dedicated to managing memory access requests from one or more peripherals. It has an arbiter for handling the priority between DMA requests.

9.2 DMA main features

- 12 independently configurable channels (requests): 7 for DMA1 and 5 for DMA2
- Each of the 12 channels is connected to dedicated hardware DMA requests, software trigger is also supported on each channel. This configuration is done by software.
- Priorities between requests from channels of one DMA are software programmable (4 levels consisting of *very high*, *high*, *medium*, *low*) or hardware in case of equality (request 1 has priority over request 2, etc.)
- Independent source and destination transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data size.
- Support for circular buffer management
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel
- Memory-to-memory transfer
- Peripheral-to-memory and memory-to-peripheral, and peripheral-to-peripheral transfers
- Access to Flash, SRAM, APB1, APB2 and AHB peripherals as source and destination
- Programmable number of data to be transferred: up to 65536

The block diagram is shown in *Figure 20* and *Figure 21*.

Figure 20. DMA block diagram in low and medium-density STM32F100xx devices

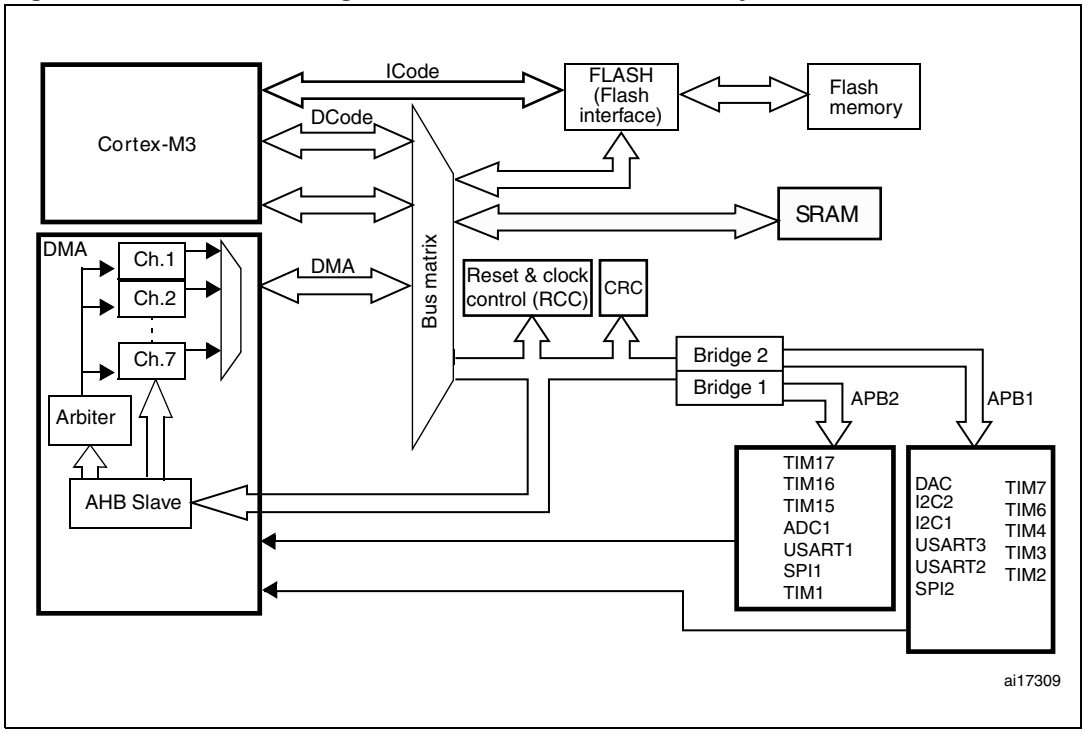
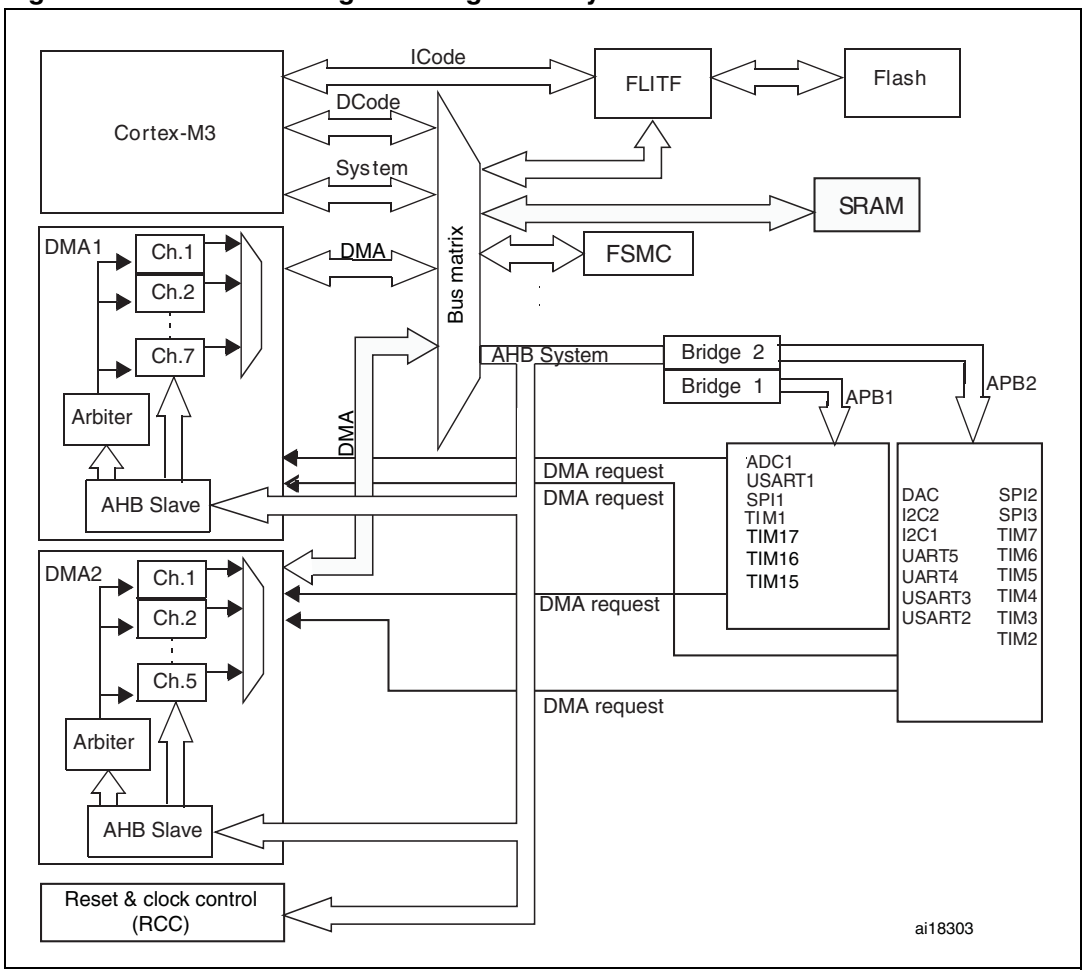


Figure 21. DMA block diagram in high-density STM32F100xx devices



Note: 1 The DMA2 controller and its related requests are available only in high-density value line devices.

2 SPI3, UART4, UART5 and TIM5 DMA requests are available only in high-density devices.

9.3 DMA functional description

The DMA controller performs direct memory transfer by sharing the system bus with the Cortex™-M3 core. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (memory or peripheral). The bus matrix implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU.

9.3.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA Controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA Controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA Controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA Controller. Once the request is deasserted by the peripheral, the DMA Controller

release the Acknowledge. If there are more requests, the peripheral can initiate the next transaction.

In summary, each DMA transfer consists of three operations:

- The loading of data from the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The storage of the data loaded to the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The post-decrementing of the DMA_CNDTRx register, which contains the number of transactions that have still to be performed.

9.3.2 Arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number will get priority versus the channel with the highest number. For example, channel 2 gets priority over channel 4.

Note: In high-density value line devices, the DMA1 controller has priority over the DMA2 controller.

9.3.3 DMA channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65535) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.

Programmable data sizes

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.

Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented after each transaction depending on the PINC and MINC bits in the DMA_CCRx register. If incremented mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPARx/DMA_CMARx registers. During transfer operations, 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 is configured in noncircular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero). In order to reload a new number of data items to be transferred into the DMA_CNDTRx register, the DMA channel must be disabled.

Note: If a DMA channel 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 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/DMA_CMARx registers.

Channel configuration procedure

The following sequence should be followed to configure a DMA channelx (where x is the channel number).

1. Set the peripheral register address in the DMA_CPARx register. The data will be moved from/ to this address to/ from the memory after the peripheral event.
2. Set the memory address in the DMA_CMARx register. The data will be written to or read from this memory after the peripheral event.
3. Configure the total number of data to be transferred in the DMA_CNDTRx register. After each peripheral event, this value will be decremented.
4. Configure the channel priority using the PL[1:0] bits in the DMA_CCRx register
5. Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register
6. Activate the channel by setting the ENABLE bit in the DMA_CCRx register.

As soon as the channel is enabled, it can serve any DMA request from the peripheral connected on the channel.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set and an interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

Circular mode

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode.

If the MEM2MEM bit in the DMA_CCRx register is set, then the channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA_CCRx register.

The transfer stops once the DMA_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode.

9.3.4 Programmable data width, data alignment and endians

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in *Table 52: Programmable data width & endian behavior (when bits PINC = MINC = 1)*.

Table 52. Programmable data width & endian behavior (when bits PINC = MINC = 1)

Source port width	Destination port width	Number of data items to transfer (NDT)	Source content: address / data	Transfer operations	Destination content: address / data
8	8	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B1[7:0] @0x1 then WRITE B1[7:0] @0x1 3: READ B2[7:0] @0x2 then WRITE B2[7:0] @0x2 4: READ B3[7:0] @0x3 then WRITE B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 00B0[15:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 00B1[15:0] @0x2 3: READ B3[7:0] @0x2 then WRITE 00B2[15:0] @0x4 4: READ B4[7:0] @0x3 then WRITE 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 000000B0[31:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 000000B1[31:0] @0x4 3: READ B3[7:0] @0x2 then WRITE 000000B2[31:0] @0x8 4: READ B4[7:0] @0x3 then WRITE 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B2[7:0] @0x1 3: READ B5B4[15:0] @0x4 then WRITE B4[7:0] @0x2 4: READ B7B6[15:0] @0x6 then WRITE B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B3B2[15:0] @0x2 3: READ B5B4[15:0] @0x4 then WRITE B5B4[15:0] @0x4 4: READ B7B6[15:0] @0x6 then WRITE B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE 0000B1B0[31:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE 0000B3B2[31:0] @0x4 3: READ B5B4[15:0] @0x4 then WRITE 0000B5B4[31:0] @0x8 4: READ B7B6[15:0] @0x6 then WRITE 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B1B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B5B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B9B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BDBC[7:0] @0x3	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B3B2B1B0[31:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B7B6B5B4[31:0] @0x4 3: READ BBBAB9B8[31:0] @0x8 then WRITE BBBAB9B8[31:0] @0x8 4: READ BFBEBDBC[31:0] @0xC then WRITE BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Addressing an AHB peripheral that does not support byte or halfword write operations

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB slave peripheral does not support byte or halfword write operations (when HSIZE is not used by the peripheral)

and does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:

- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it will transform any AHB byte or halfword operation into a 32-bit APB operation in the following manner:

- an AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0
- an AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) will be converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0

For instance, if you want to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), you must configure the memory source size (MSIZE) to “16-bit” and the peripheral destination size (PSIZE) to “32-bit”.

9.3.5 Error management

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding Channel configuration register (DMA_CCRx). The channel's transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA_CCRx register is set.

9.3.6 Interrupts

An interrupt can be produced on a Half-transfer, Transfer complete or Transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.

Table 53. DMA interrupt requests

Interrupt event	Event flag	Enable Control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

Note: In high-density value line devices, DMA2 Channel4 and DMA2 Channel5 interrupts are mapped onto the same interrupt vector. All other DMA1 and DMA2 Channel interrupts have their own interrupt vector.

9.3.7 DMA request mapping

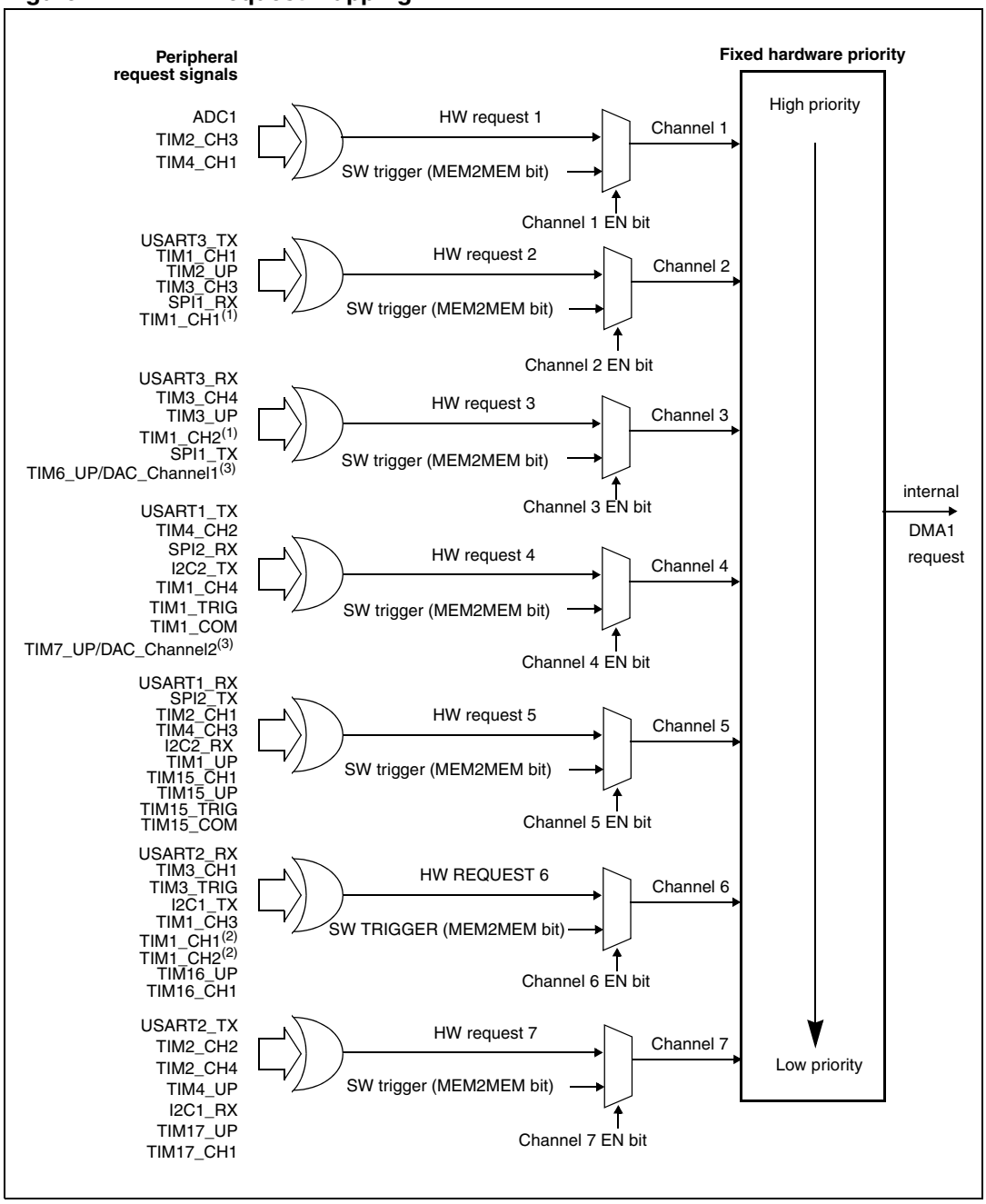
DMA1 controller

The 7 requests from the peripherals (TIMx[1,2,3,4,6,7,15,16,17], ADC1, SPI[1,2], I2Cx[1,2], USARTx[1,2,3]) and DAC Channelx[1,2] are simply logically ORed before entering the

DMA1, this means that only one request must be enabled at a time. Refer to *Figure 22: DMA1 request mapping*.

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.

Figure 22. DMA1 request mapping



1. The TIM1_CH1 and TIM1_CH2 DMA requests are mapped on DMA Channel 2 and DMA Channel 3, respectively, only if the TIM1_DMA_REMAP bit in the AFIO_MAPR2 register is cleared. For more details please refer to the AFIO section.
2. The TIM1_CH1 and TIM1_CH2 DMA requests are mapped on DMA Channel 6 only if the TIM1_DMA_REMAP bit in the AFIO_MAPR2 register is set. For more details please refer to the AFIO

section.

3. For High-density value line devices, the TIM6_DAC1 and TIM7_DAC2 DMA requests are mapped respectively on DMA1 Channel 3 and DMA1 Channel 4 only if the TIM67_DAC_DMA_REMAP bit in the AFIO_MAPR2 register is set and mapped respectively on DMA2 Channel 3 and DMA2 Channel 4 when the TIM67_DAC_DMA_REMAP bit in the AFIO_MAPR2 register is reset.
On low- and medium -density devices the TIM6_DAC1 and TIM7_DAC2 DMA requests are always mapped respectively on DMA1 Channel 3 and DMA1 Channel 4. For more details please refer to the AFIO section.

Table 54 lists the DMA requests for each channel.

Table 54. Summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1						
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3 TIM1_CH2 TIM1_CH1	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP
TIM6/DAC_Channel1			TIM6_UP/DAC_Channel1				
TIM7/DAC_Channel2				TIM7_UP/DAC_Channel2			
TIM15					TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM		
TIM16						TIM16_CH1 TIM16_UP	
TIM17							TIM17_CH1 TIM17_UP

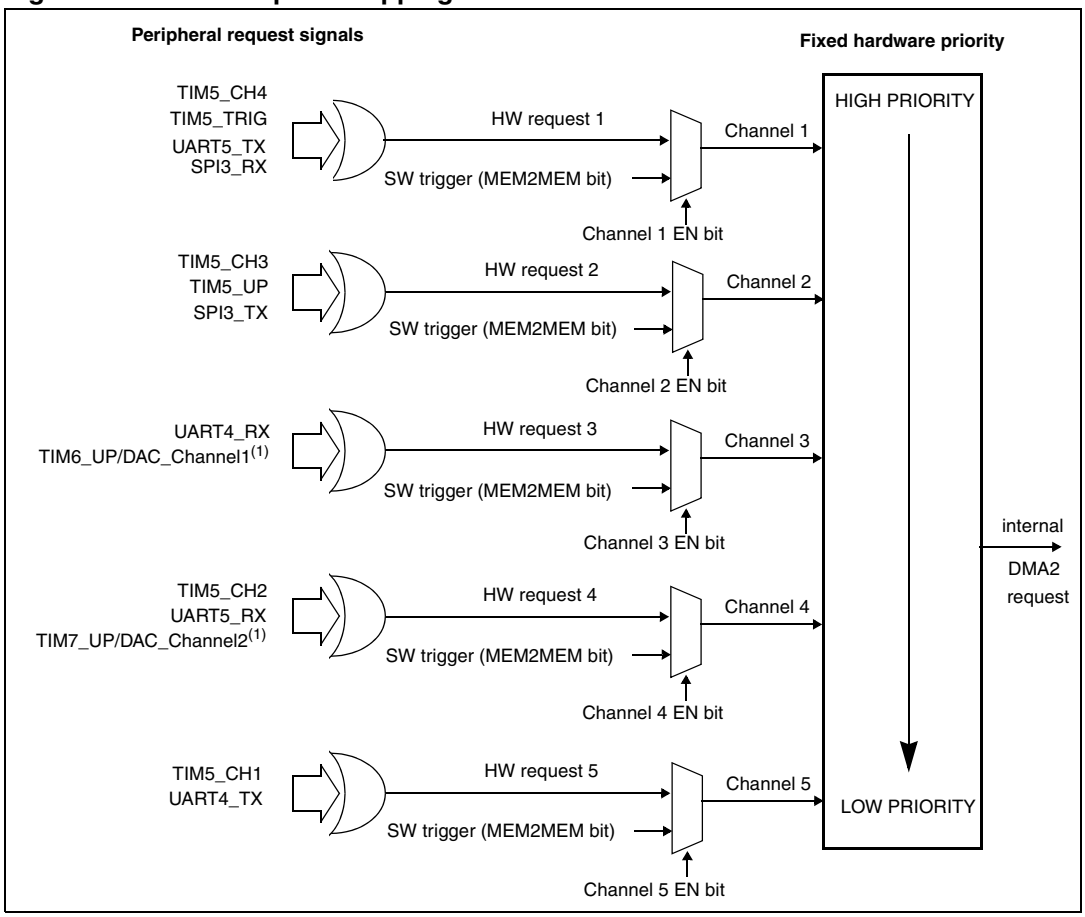
DMA2 controller

The 5 requests from the peripherals (TIMx[5,6,7], SPI3, UARTx[4,5], DAC_Channel[1,2]) are simply logically ORed before entering the DMA2, this means that only one request must be enabled at a time. Refer to *Figure 23: DMA2 request mapping*.

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.

Note: The DMA2 controller and its relative requests are available only in high-density value line devices.

Figure 23. DMA2 request mapping



1. For high-density value line devices, the TIM6_DAC1 and TIM7_DAC2 DMA requests are mapped respectively on DMA1 Channel 3 and DMA1 Channel 4 only if the TIM67_DAC_DMA_REMAP bit in the AFIO_MAPR2 register is set, and mapped respectively on DMA2 Channel 3 and DMA2 Channel 4 when the TIM67_DAC_DMA_REMAP bit in the AFIO_MAPR2 register is reset. On low- and medium -density devices the TIM6_DAC1 and TIM7_DAC2 DMA requests are always mapped respectively on DMA1 Channel 3 and DMA1 Channel 4. For more details please refer to the AFIO section.

Table 55 lists the DMA2 requests for each channel.

Table 55. Summary of DMA2 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
SPI3	SPI3_RX	SPI3_TX			
UART4			UART4_RX		UART4_TX
UART5	UART5_TX			UART5_RX	
TIM5	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
TIM6/ DAC_Channel1			TIM6_UP/ DAC_Channel1		
TIM7/ DAC_Channel2				TIM7_UP/ DAC_Channel2	

9.4 DMA registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

Note: In the following registers, all bits related to channel6 and channel7 are not relevant for DMA2 since it has only 5 channels.

The peripheral registers can be accessed by bytes (8-bit), half-words (16-bit) or words (32-bit).

9.4.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 27, 23, 19, 15, 11, 7, 3 **TEIFx:** Channel x transfer error flag (x = 1 ..7)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No transfer error (TE) on channel x
1: A transfer error (TE) occurred on channel x

Bits 26, 22, 18, 14, 10, 6, 2 **HTIFx:** Channel x half transfer flag (x = 1 ..7)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No half transfer (HT) event on channel x
1: A half transfer (HT) event occurred on channel x

Bits 25, 21, 17, 13, 9, 5, 1 **TCIFx:** Channel x transfer complete flag (x = 1 ..7)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No transfer complete (TC) event on channel x
1: A transfer complete (TC) event occurred on channel x

Bits 24, 20, 16, 12, 8, 4, 0 **GIFx:** Channel x global interrupt flag (x = 1 ..7)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.
0: No TE, HT or TC event on channel x
1: A TE, HT or TC event occurred on channel x

9.4.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 27, 23, 19, 15, 11, 7, 3 **CTEIFx:** Channel x transfer error clear (x = 1 ..7)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding TEIF flag in the DMA_ISR register

Bits 26, 22, 18, 14, 10, 6, 2 **CHTIFx:** Channel x half transfer clear (x = 1 ..7)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding HTIF flag in the DMA_ISR register

Bits 25, 21, 17, 13, 9, 5, 1 **CTCIFx:** Channel x transfer complete clear (x = 1 ..7)
This bit is set and cleared by software.
0: No effect
1: Clears the corresponding TCIF flag in the DMA_ISR register

Bits 24, 20, 16, 12, 8, 4, 0 **CGIFx:** Channel x global interrupt clear (x = 1 ..7)
This bit is set and cleared by software.
0: No effect
1: Clears the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

9.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1..7, where x = channel number)

Address offset: 0x08 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **MEM2MEM:** Memory to memory mode
This bit is set and cleared by software.
0: Memory to memory mode disabled
1: Memory to memory mode enabled

Bits 13:12 **PL[1:0]:** Channel priority level
These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high

Bits 11:10 **MSIZE[1:0]:** Memory size
These bits are set and cleared by software.
00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bits 9:8 **PSIZE[1:0]:** Peripheral size
These bits are set and cleared by software.
00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bit 7 **MINC:** Memory increment mode
This bit is set and cleared by software.
0: Memory increment mode disabled
1: Memory increment mode enabled

Bit 6 **PINC:** Peripheral increment mode
This bit is set and cleared by software.
0: Peripheral increment mode disabled
1: Peripheral increment mode enabled

Bit 5 **CIRC:** Circular mode
This bit is set and cleared by software.
0: Circular mode disabled
1: Circular mode enabled

Bit 4 **DIR:** Data transfer direction
This bit is set and cleared by software.
0: Read from peripheral
1: Read from memory

Bit 3 **TEIE:** Transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 2 **HTIE:** Half transfer interrupt enable
This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 1 **TCIE:** Transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 0 **EN:** Channel enable
This bit is set and cleared by software.
0: Channel disabled
1: Channel enabled

9.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1..7), where x = channel number)

Address offset: 0x0C + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT															
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 **NDT[15:0]:** Number of data to transfer
Number of data to be transferred (0 up to 65535). This register can only be written when the channel is disabled. Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer.

Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in auto-reload mode.

If this register is zero, no transaction can be served whether the channel is enabled or not.

9.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1..7), where x = channel number)

Address offset: 0x10 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

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
PA																															
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	rw

Bits 31:0 **PA[31:0]:** Peripheral address

Base address of the peripheral data register from/to which the data will be read/written.

When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half-word address.
When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.

9.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1..7), where x = channel number)

Address offset: 0x14 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

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
MA																															
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	rw

Bits 31:0 **MA[31:0]:** Memory address

Base address of the memory area from/to which the data will be read/written.

When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address.
When MSIZE is 10 (32-bit), MA[1:0] are ignored. Access is automatically aligned to a word address.

9.4.7 DMA register map

The following table gives the DMA register map and the reset values.

Table 56. DMA 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	**DMA_ISR**	Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	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
0x004	**DMA_IFCR**	Reserved				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	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
0x008	**DMA_CCR1**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**DMA_CNDTR1**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**DMA_CPAR1**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**DMA_CMAR1**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved																																
0x01C	**DMA_CCR2**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**DMA_CNDTR2**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**DMA_CPAR2**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**DMA_CMAR2**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved																																
0x030	**DMA_CCR3**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	**DMA_CNDTR3**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**DMA_CPAR3**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**DMA_CMAR3**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved																																
0x044	**DMA_CCR4**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	**DMA_CNDTR4**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 56. DMA 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
0x04C	**DMA_CPAR4**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	**DMA_CMAR4**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved																																
0x058	**DMA_CCR5**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	**DMA_CNDTR5**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	**DMA_CPAR5**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	**DMA_CMAR5**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	**DMA_CCR6**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**DMA_CNDTR6**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	**DMA_CPAR6**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	**DMA_CMAR6**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved																																
0x080	**DMA_CCR7**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**DMA_CNDTR7**	Reserved																	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	**DMA_CPAR7**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	**DMA_CMAR7**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	Reserved																																

Refer to for the register boundary addresses.

10 Analog-to-digital converter (ADC)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This Section applies to the whole STM32F100xx family, unless otherwise specified.

10.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 18 multiplexed channels allowing it measure signals from 16 external and two internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined high or low thresholds.

The ADC input clock is generated from the PCLK2 clock divided by a prescaler , refer to *Figure 8: STM32F100xx clock tree (low and medium-density devices)* and *Figure 9: STM32F100xx clock tree (high-density devices)*.

10.2 ADC main features

- 12-bit resolution
- Interrupt generation at End of Conversion, End of Injected conversion and Analog watchdog event
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel ‘n’
- Self-calibration
- Data alignment with in-built data coherency
- Channel by channel programmable sampling time
- External trigger option for both regular and injected conversion
- Discontinuous mode
- ADC conversion time:
  - STM32F100xx value line devices: 1.17 µs at 24 MHz
- ADC supply requirement: 2.4 V to 3.6 V
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- DMA request generation during regular channel conversion

The block diagram of the ADC is shown in *Figure 24*.

Note: V_{REF-}, if available (depending on package), must be tied to V_{SSA}.

10.3 ADC functional description

Figure 24 shows a single ADC block diagramand *Table 57* gives the ADC pin description.

Figure 24. Single ADC block diagram

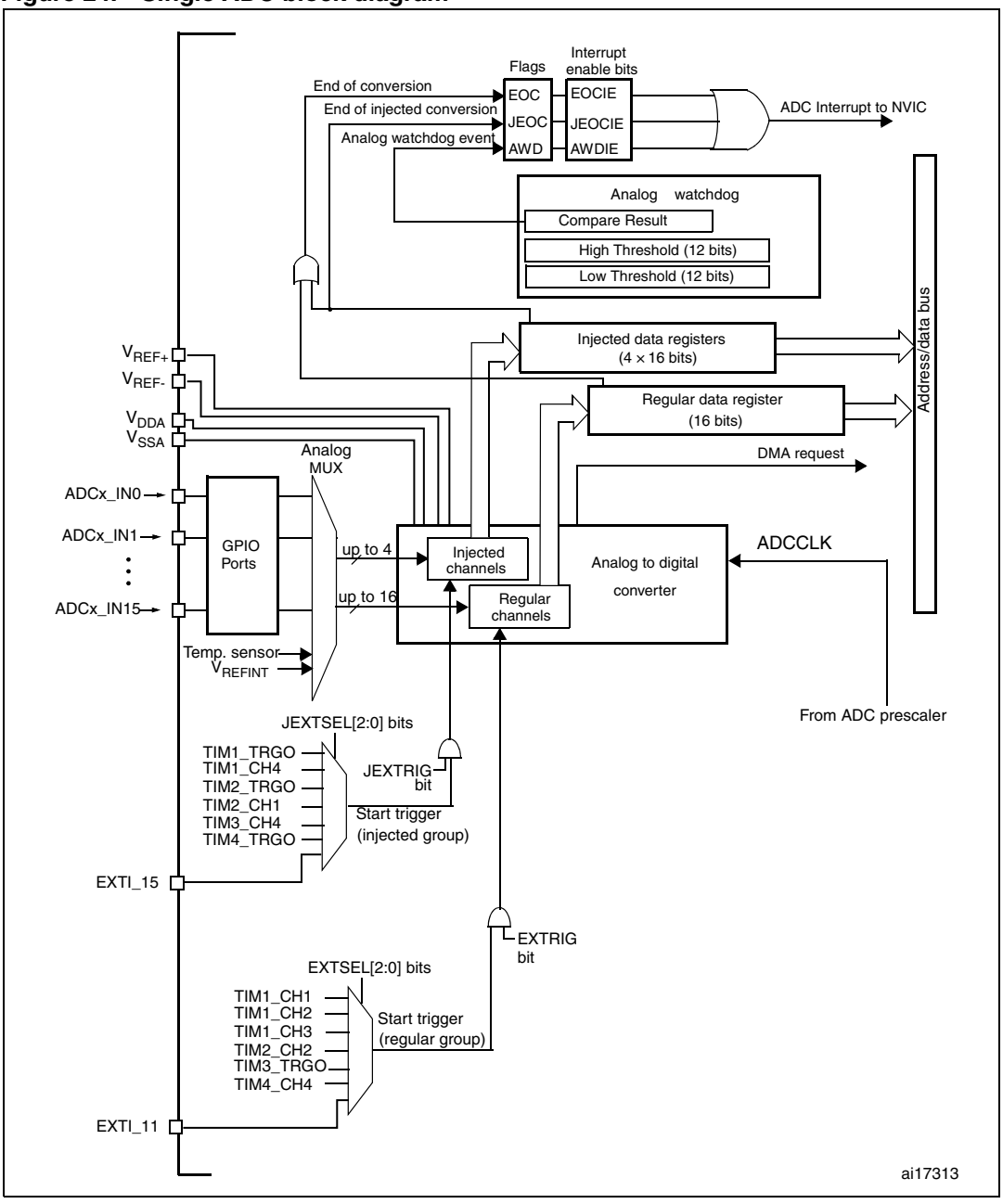


Table 57. ADC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, 2.4 V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}(1)	Input, analog supply	Analog power supply equal to V_{DD} and 2.4 V ≤ V_{DDA} ≤ 3.6 V
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, V_{REF-} = V_{SSA}
V_{SSA}(1)	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADCx_IN[15:0]	Analog signals	Up to 21 analog channels(2)

1. V_{DDA} and V_{SSA} have to be connected to V_{DD} and V_{SS}, respectively.
2. For full details about the ADC I/O pins, please refer to the “Pinouts and pin descriptions” section of the corresponding device datasheet.

10.3.1 ADC on-off control

The ADC can be powered-on by setting the ADON bit in the ADC_CR2 register. When the ADON bit is set for the first time, it wakes up the ADC from Power Down mode.

Conversion starts when ADON bit is set for a second time by software after ADC power-up time (t_{STAB}).

You can stop conversion and put the ADC in power down mode by resetting the ADON bit. In this mode the ADC consumes almost no power (only a few µA).

10.3.2 ADC clock

The ADCCLK clock provided by the Clock Controller is synchronous with the PCLK2 (APB2 clock). The RCC controller has a dedicated programmable prescaler for the ADC clock (refer to *Section 6: Reset and clock control (RCC)*for more details.

10.3.3 Channel selection

There are 16 multiplexed channels. It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions which can be done on any channel and in any order. For instance, it is possible to do the conversion in the following order: Ch3, Ch8, Ch2, Ch2, Ch0, Ch2, Ch2, Ch15.

- The **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_SQRx registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- The **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.

If the ADC_SQRx or ADC_JSQR registers are modified during a conversion, the current conversion is reset and a new start pulse is sent to the ADC to convert the new chosen group.

Temperature sensor/V_{REFINT} internal channels

The Temperature sensor is connected to channel ADCx_IN16 and the internal reference voltage V_{REFINT} is connected to ADCx_IN17. These two internal channels can be selected and converted as injected or regular channels.

Note: The sensor and V_{REFINT} are only available on the master ADC1 peripheral.

10.3.4 Single conversion mode

In Single conversion mode the ADC does one conversion. This mode is started either by setting the ADON bit in the ADC_CR2 register (for a regular channel only) or by external trigger (for a regular or injected channel), while the CONT bit is 0.

Once the conversion of the selected channel is complete:

- If a regular channel was converted:
  - The converted data is stored in the 16-bit ADC_DR register
  - The EOC (End Of Conversion) flag is set
  - and an interrupt is generated if the EOCIE is set.
- If an injected channel was converted:
  - The converted data is stored in the 16-bit ADC_DRJ1 register
  - The JEOC (End Of Conversion Injected) flag is set
  - and an interrupt is generated if the JEOCIE bit is set.

The ADC is then stopped.

10.3.5 Continuous conversion mode

In continuous conversion mode ADC starts another conversion as soon as it finishes one. This mode is started either by external trigger or by setting the ADON bit in the ADC_CR2 register, while the CONT bit is 1.

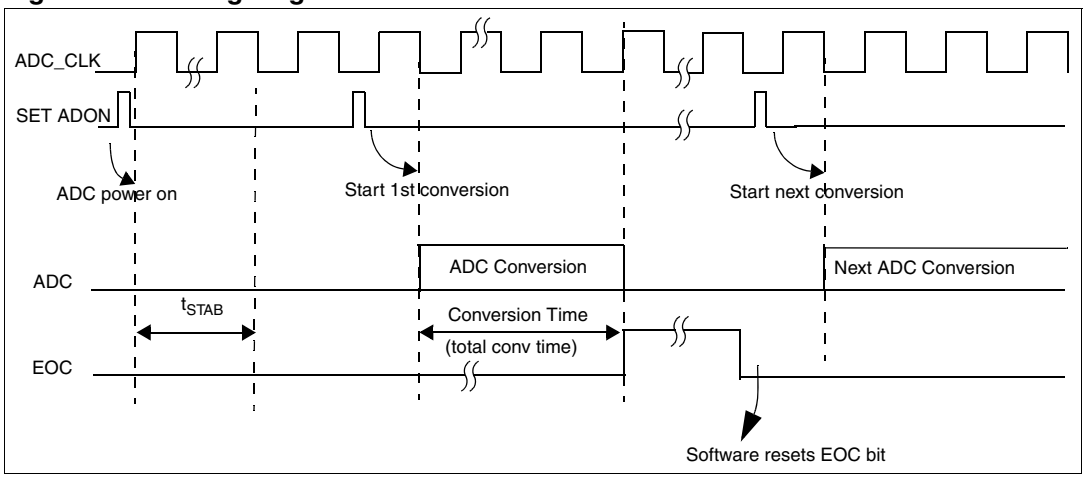
After each conversion:

- If a regular channel was converted:
  - The converted data is stored in the 16-bit ADC_DR register
  - The EOC (End Of Conversion) flag is set
  - An interrupt is generated if the EOCIE is set.
- If an injected channel was converted:
  - The converted data is stored in the 16-bit ADC_DRJ1 register
  - The JEOC (End Of Conversion Injected) flag is set
  - An interrupt is generated if the JEOCIE bit is set.

10.3.6 Timing diagram

As shown in *Figure 25*, the ADC needs a stabilization time of t_{STAB} before it starts converting accurately. After the start of ADC conversion and after 14 clock cycles, the EOC flag is set and the 16-bit ADC Data register contains the result of the conversion.

Figure 25. Timing diagram



10.3.7 Analog watchdog

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in the 12 least significant bits of the ADC_HTR and ADC_LTR 16-bit registers. An interrupt can be enabled by using the AWDIE bit in the ADC_CR1 register.

The threshold value is independent of the alignment selected by the ALIGN bit in the ADC_CR2 register. The comparison is done before the alignment (see *Section 10.5*).

The analog watchdog can be enabled on one or more channels by configuring the ADC_CR1 register as shown in *Table 58.*

Figure 26. Analog watchdog guarded area

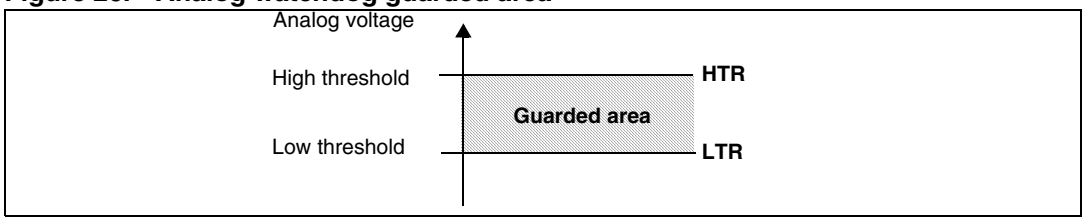


Table 58. Analog watchdog channel selection

Channels to be guarded by analog watchdog	ADC_CR1 register control bits (x = don't care)		
	AWDSGL bit	AWDEN bit	JAWDEN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by AWDCH[4:0] bits

10.3.8 Scan mode

This mode is used to scan a group of analog channels.

Scan mode can be selected by setting the SCAN bit in the ADC_CR1 register. Once this bit is set, ADC scans all the channels selected in the ADC_SQRx registers (for regular channels) or in the ADC_JSQR (for injected channels). A single conversion is performed for each channel of the group. After each end of conversion the next channel of the group is converted automatically. If the CONT bit is set, conversion does not stop at the last selected group channel but continues again from the first selected group channel.

If the DMA bit is set, the direct memory access controller is used to transfer the converted data of regular group channels to SRAM after each EOC.

The injected channel converted data is always stored in the ADC_JDRx registers.

10.3.9 Injected channel management

Triggered injection

To use triggered injection, the JAUTO bit must be cleared and SCAN bit must be set in the ADC_CR1 register.

1. Start conversion of a group of regular channels either by external trigger or by setting the ADON bit in the ADC_CR2 register.
2. If an external injected trigger occurs during the regular group channel conversion, the current conversion is reset and the injected channel sequence is converted in Scan once mode.
3. Then, the regular group channel conversion is resumed from the last interrupted regular conversion. If a regular event occurs during an injected conversion, it doesn't interrupt it but the regular sequence is executed at the end of the injected sequence. *Figure 27* shows the 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 28 ADC clock cycles (that is two conversions with a 1.5 clock-period sampling time), the minimum interval between triggers must be 29 ADC clock cycles.

Auto-injection

If the JAUTO bit is set, then the injected group channels are automatically converted after the regular group channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRx and ADC_JSQR registers.

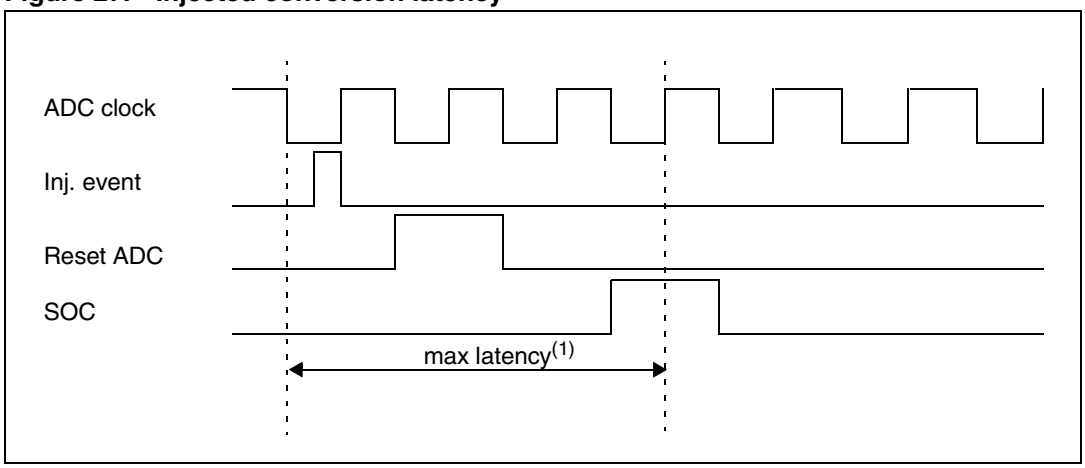
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

For ADC clock prescalers ranging from 4 to 8, a delay of 1 ADC clock period is automatically inserted when switching from regular to injected sequence (respectively injected to regular). When the ADC clock prescaler is set to 2, the delay is 2 ADC clock periods.

Note: It is not possible to use both auto-injected and discontinuous modes simultaneously.

Figure 27. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32F101xx and STM32F103xx datasheets.

10.3.10 Discontinuous mode

Regular group

This mode is enabled by setting the DISCEN bit in the ADC_CR1 register. It can be used to convert a short sequence of n conversions (n <=8) which is a part of the sequence of conversions selected in the ADC_SQRx registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRx 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:

n = 3, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10
1st trigger: sequence converted 0, 1, 2
2nd trigger: sequence converted 3, 6, 7
3rd trigger: sequence converted 9, 10 and an EOC event generated
4th trigger: sequence converted 0, 1, 2

Note: When a regular group is converted in discontinuous mode, no rollover will occur. When all sub groups are converted, the next trigger starts conversion of the first sub-group.

In the example above, the 4th trigger reconverts the 1st sub-group channels 0, 1 and 2.

Injected group

This mode is enabled by setting the JDISCEN bit in the ADC_CR1 register. It can be used to convert the sequence selected in the ADC_JSQR register, channel by channel, after an external trigger event.

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:

n = 1, channels to be converted = 1, 2, 3
1st trigger: channel 1 converted
2nd trigger: channel 2 converted
3rd trigger: channel 3 converted and EOC and JEOC events generated
4th trigger: channel 1

Note: 1 When all injected channels are 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.

2 It is not possible to use both auto-injected and discontinuous modes simultaneously.

3 The user must avoid setting discontinuous mode for both regular and injected groups together. Discontinuous mode must be enabled only for one group conversion.

10.4 Calibration

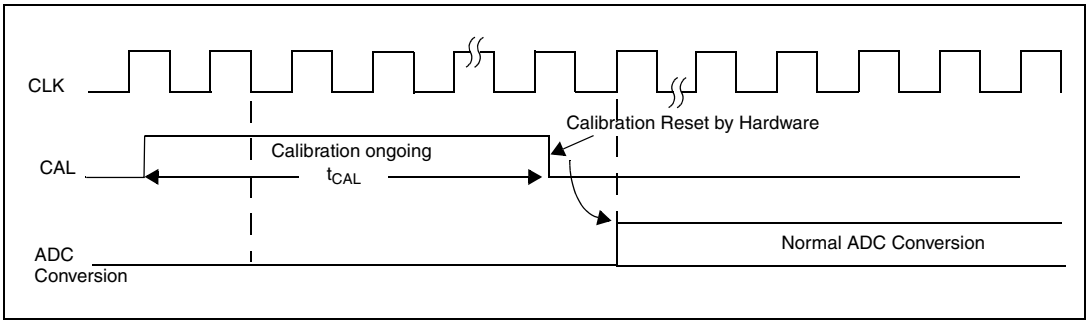
The ADC has an built-in self calibration mode. Calibration significantly reduces accuracy errors due to internal capacitor bank variations. During calibration, an error-correction code (digital word) is calculated for each capacitor, and during all subsequent conversions, the error contribution of each capacitor is removed using this code.

Calibration is started by setting the CAL bit in the ADC_CR2 register. Once calibration is over, the CAL bit is reset by hardware and normal conversion can be performed. It is recommended to calibrate the ADC once at power-on. The calibration codes are stored in the ADC_DR as soon as the calibration phase ends.

Note: 1 It is recommended to perform a calibration after each power-up.

2 Before starting a calibration the ADC must have been in power-off state (ADON bit = '0') for at least two ADC clock cycles.

Figure 28. Calibration timing diagram



10.5 Data alignment

ALIGN bit in the ADC_CR2 register selects the alignment of data stored after conversion. Data can be left or right aligned as shown in *Figure 29.* and *Figure 30.*

The injected group channels converted data value is decreased by the user-defined offset written in the ADC_JOFRx registers so the result can be a negative value. The SEXT bit is the extended sign value.

For regular group channels no offset is subtracted so only twelve bits are significant.

Figure 29. Right alignment of data

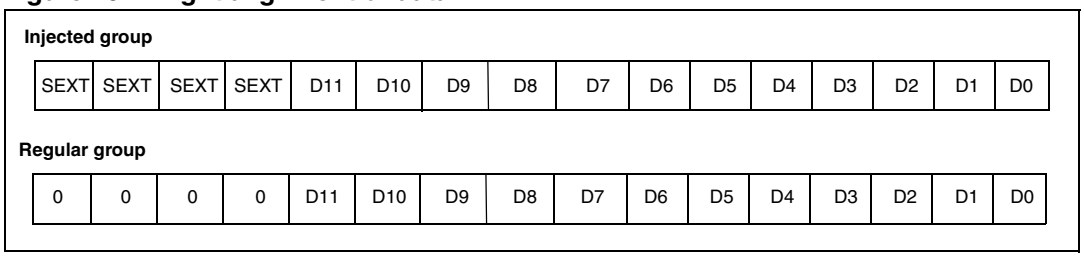
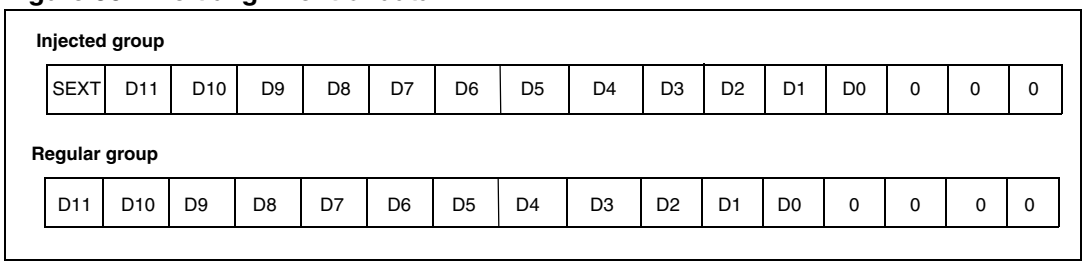


Figure 30. Left alignment of data



10.6 Channel-by-channel programmable sample time

ADC samples the input voltage for a number of ADC_CLK cycles which can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sample time.

The total conversion time is calculated as follows:

Tconv = Sampling time + 12.5 cycles

Example:

With an ADCCLK = 12 MHz and a sampling time of 1.5 cycles:

Tconv = 1.5 + 12.5 = 14 cycles = 1.17 µs

10.7 Conversion on external trigger

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the EXTTRIG control bit is set then external events are able to trigger a conversion. The EXTSEL[2:0] and JEXTSEL[2:0] control bits allow the application to select decide which out of 8 possible events can trigger conversion for the regular and injected groups.

Note: When an external trigger is selected for ADC regular or injected conversion, only the rising edge of the signal can start the conversion.

Table 59. External trigger for regular channels for ADC1

Source	Type	EXTSEL[2:0]
TIM1_CC1 event	Internal signal from on-chip timers	000
TIM1_CC2 event		001
TIM1_CC3 event		010
TIM2_CC2 event		011
TIM3_TRGO event		100
TIM4_CC4 event		101
EXTI line 11	External pin	110
SWSTART	Software control bit	111

Table 60. External trigger for injected channels for ADC1

Source	Connection type	JEXTSEL[2:0]
TIM1_TRGO event	Internal signal from on-chip timers	000
TIM1_CC4 event		001
TIM2_TRGO event		010
TIM2_CC1 event		011
TIM3_CC4 event		100
TIM4_TRGO event		101
EXTI line 15	External pin	110
JSWSTART	Software control bit	111

The software source trigger events can be generated by setting a bit in a register (SWSTART and JSWSTART in ADC_CR2).

A regular group conversion can be interrupted by an injected trigger.

10.8 DMA request

Since converted regular channels value are stored in a unique data register, it is necessary to use DMA for conversion of more than one regular channel. This avoids the loss of data already stored in the ADC_DR register.

Only the end of conversion of a regular channel generates a DMA request, which allows the transfer of its converted data from the ADC_DR register to the destination location selected by the user.

10.9 Temperature sensor

The temperature sensor can be used to measure the ambient temperature (T_A) of the device.

The temperature sensor is internally connected to the ADCx_IN16 input channel which is used to convert the sensor output voltage into a digital value. The recommended sampling time for the temperature sensor is 17.1 µs.

The block diagram of the temperature sensor is shown in *Figure 31*.

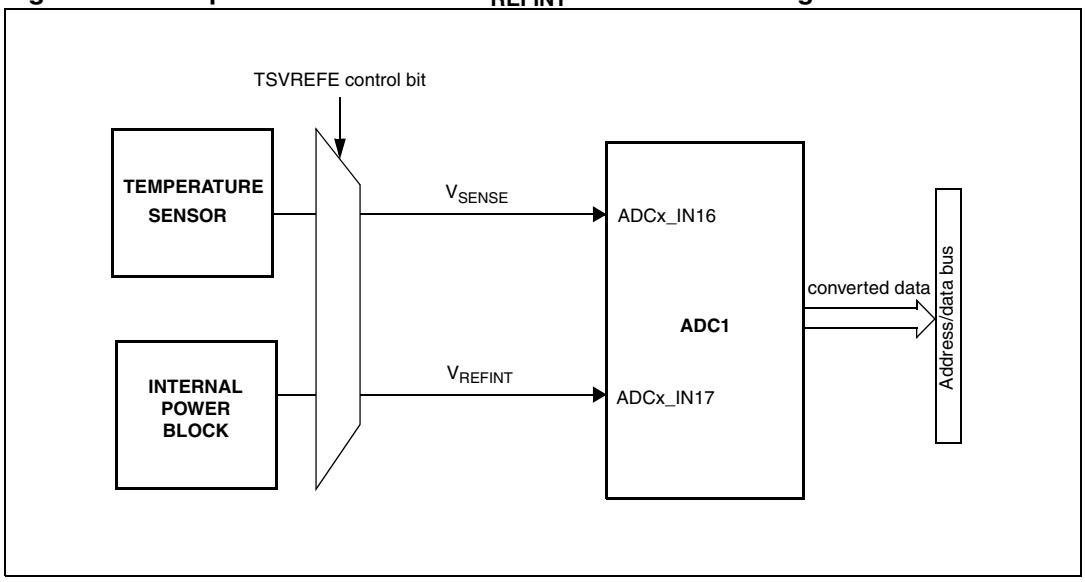
When not in use, this sensor can be put in power down mode.

Note: The TSVREFE bit must be set to enable both internal channels: ADCx_IN16 (temperature sensor) and ADCx_IN17 (V_{REFINT}) conversion.

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 internal temperature sensor is more suited to applications that detect temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor part should be used.

Figure 31. Temperature sensor and V_{REFINT} channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADCx_IN16 input channel.
2. Select a sample time of 17.1 µs
3. Set the TSVREFE bit in the *ADC control register 2 (ADC_CR2)* to wake up the temperature sensor from power down mode.
4. Start the ADC conversion by setting the ADON bit (or by external trigger).
5. Read the resulting V_{SENSE} data in the ADC data register
6. Obtain the temperature using the following formula:

   Temperature (in °C) = {(V_{25} - V_{SENSE}) / Avg_Slope} + 25.

   Where,

   V_{25} = V_{SENSE} value for 25° C and

   Avg_Slope = Average Slope for curve between Temperature vs. V_{SENSE} (given in mV/° C or µV/ °C).

   Refer to the Electrical characteristics section for the actual values of V_{25} and Avg_Slope.

Note: The sensor has a startup time after waking from power down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADON and TSVREFE bits should be set at the same time.

10.10 ADC interrupts

An interrupt can be produced on end of conversion for regular and injected groups and when the analog watchdog status bit is set. Separate interrupt enable bits are available for flexibility.

Two other flags are present in the ADC_SR register, but there is no interrupt associated with them:

- JSTRT (Start of conversion for injected group channels)
- STRT (Start of conversion for regular group channels)

Table 61. ADC interrupts

Interrupt event	Event flag	Enable Control bit
End of conversion regular group	EOC	EOCIE
End of conversion injected group	JEOC	JEOCIE
Analog watchdog status bit is set	AWD	AWDIE

10.11 ADC registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

10.11.1 ADC status register (ADC_SR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											STRT	JSTRT	JEOC	EOC	AWD
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **STRT:** Regular channel Start flag

This bit is set by hardware when regular channel conversion starts. It is cleared by software.

0: No regular channel conversion started

1: Regular channel conversion has started

Bit 3 **JSTRT:** Injected channel Start flag

This bit is set by hardware when injected channel group conversion starts. It is cleared by software.

0: No injected group conversion started

1: Injected group conversion has started

Bit 2 **JEOC:** Injected channel end of conversion

This bit is set by hardware at the end of all injected group channel conversion. It is cleared by software.

0: Conversion is not complete

1: Conversion complete

Bit 1 **EOC:** End of conversion

This bit is set by hardware at the end of a group channel conversion (regular or injected). It is cleared by software or by reading the ADC_DR.

0: Conversion is not complete

1: Conversion complete

Bit 0 **AWD:** Analog watchdog flag

This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software.

0: No Analog watchdog event occurred

1: Analog watchdog event occurred

10.11.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								AWDEN	JAWDEN	Reserved					
								rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOC IE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **AWDEN:** Analog watchdog enable on regular channels
This bit is set/reset by software.
0: Analog watchdog disabled on regular channels
1: Analog watchdog enabled on regular channels

Bit 22 **JAWDEN:** Analog watchdog enable on injected channels
This bit is set/reset by software.
0: Analog watchdog disabled on injected channels
1: Analog watchdog enabled on injected channels

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:13 **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

Bit 12 **JDISCEN**: Discontinuous mode on injected channels
This bit set and cleared by software to enable/disable discontinuous mode on injected group channels
0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled

Bit 11 **DISCEN**: Discontinuous mode on regular channels
This bit set and cleared by software to enable/disable Discontinuous mode on regular channels.
0: Discontinuous mode on regular channels disabled
1: Discontinuous mode on regular channels enabled

Bit 10 **JAUTO:** Automatic Injected Group conversion
This bit set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.
0: Automatic injected group conversion disabled
1: Automatic injected group conversion enabled

Bit 9 **AWDSGL:** Enable the watchdog on a single channel in scan mode

This bit set and cleared by software to enable/disable the analog watchdog on the channel identified by the AWDCH[4:0] bits.

0: Analog watchdog enabled on all channels

1: Analog watchdog enabled on a single channel

Bit 8 **SCAN:** Scan mode

This bit is set and cleared by software to enable/disable Scan mode. In Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted.

0: Scan mode disabled

1: Scan mode enabled

Note: An EOC or JEOC interrupt is generated only on the end of conversion of the last channel if the corresponding EOCIE or JEOCIE bit is set

Bit 7 **JEOCIE**: Interrupt enable for injected channels

This bit is set and cleared by software to enable/disable the end of conversion interrupt for injected channels.

0: JEOC interrupt disabled

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Bit 6 **AWDIE***:* Analog watchdog interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt. In Scan mode if the watchdog thresholds are crossed, scan is aborted only if this bit is enabled.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Bit 5 **EOCIE:** Interrupt enable for EOC

This bit is set and cleared by software to enable/disable the End of Conversion interrupt.

0: EOC interrupt disabled

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Bits 4:0 **AWDCH[4:0]:** Analog watchdog channel select bits

These bits are set and cleared by software. They select the input channel to be guarded by the Analog watchdog.

00000: ADC analog Channel0

00001: ADC analog Channel1

....

01111: ADC analog Channel15

10000: ADC analog Channel16

10001: ADC analog Channel17

Other values reserved.

ADC1 analog Channel16 and Channel17 are internally connected to the temperature sensor and to V_{REFINT}, respectively.

10.11.3 ADC control register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL[2:0]			Res.
								rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL[2:0]			ALIGN	Reserved		DMA	Reserved				RSTCAL	CAL	CONT	ADON
rw	rw	rw	rw	rw	Res.		rw					rw	rw	rw	rw

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

Bit 23 **TSVREFE**: Temperature sensor and V_{REFINT} enable

This bit is set and cleared by software to enable/disable the temperature sensor and V_{REFINT} channel.

0: Temperature sensor and V_{REFINT} channel disabled

1: Temperature sensor and V_{REFINT} channel enabled

Bit 22 **SWSTART**: Start conversion of regular channels

This bit is set by software to start conversion and cleared by hardware as soon as conversion starts. It starts a conversion of a group of regular channels if SWSTART is selected as trigger event by the EXTSEL[2:0] bits.

0: Reset state

1: Starts conversion of regular channels

Bit 21 **JSWSTART**: Start conversion of injected channels

This bit is set by software and cleared by software or by hardware as soon as the conversion starts. It starts a conversion of a group of injected channels (if JSWSTART is selected as trigger event by the JEXTSEL[2:0] bits.

0: Reset state

1: Starts conversion of injected channels

Bit 20 **EXTTRIG**: External trigger conversion mode for regular channels

This bit is set and cleared by software to enable/disable the external trigger used to start conversion of a regular channel group.

0: Conversion on external event disabled

1: Conversion on external event enabled

Bits 19:17 **EXTSEL[2:0]**: External event select for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

000: Timer 1 CC1 event

001: Timer 1 CC2 event

010: Timer 1 CC3 event

011: Timer 2 CC2 event

100: Timer 3 TRGO event

101: Timer 4 CC4 event

110: EXTI line 11

111: SWSTART

Bit 16 Reserved, must be kept at reset value.

Bit 15 **JEXTTRIG**: External trigger conversion mode for injected channels

This bit is set and cleared by software to enable/disable the external trigger used to start conversion of an injected channel group.

0: Conversion on external event disabled

1: Conversion on external event enabled

Bits 14:12 **JEXTSEL[2:0]**: External event select for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

000: Timer 1 TRGO event
001: Timer 1 CC4 event
010: Timer 2 TRGO event
011: Timer 2 CC1 event
100: Timer 3 CC4 event
101: Timer 4 TRGO event
110: EXTI line15
111: JSWSTART

Bit 11 **ALIGN**: Data alignment

This bit is set and cleared by software. Refer to *Figure 29.*and *Figure 30.*

0: Right Alignment
1: Left Alignment

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **DMA**: Direct memory access mode

This bit is set and cleared by software. Refer to the DMA controller chapter for more details.

0: DMA mode disabled
1: DMA mode enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **RSTCAL:** Reset calibration

This bit is set by software and cleared by hardware. It is cleared after the calibration registers are initialized.

0: Calibration register initialized.
1: Initialize calibration register.

Note: If RSTCAL is set when conversion is ongoing, additional cycles are required to clear the calibration registers.

Bit 2 **CAL:** A/D Calibration

This bit is set by software to start the calibration. It is reset by hardware after calibration is complete.

0: Calibration completed
1: Enable calibration

Bit 1 **CONT:** Continuous conversion

This bit is set and cleared by software. If set conversion takes place continuously till this bit is reset.

0: Single conversion mode
1: Continuous conversion mode

Bit 0 **ADON**: A/D converter ON / OFF

This bit is set and cleared by software. If this bit holds a value of zero and a 1 is written to it then it wakes up the ADC from Power Down state.

Conversion starts when this bit holds a value of 1 and a 1 is written to it. The application should allow a delay of t_{STAB} between power up and start of conversion. Refer to *Figure 25.*

0: Disable ADC conversion/calibration and go to power down mode.
1: Enable ADC and to start conversion

Note: If any other bit in this register apart from ADON is changed at the same time, then conversion is not triggered. This is to prevent triggering an erroneous conversion.

10.11.4 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **SMPx[2:0]:** Channel x Sample time selection

These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged.

000: 1.5 cycles

001: 7.5 cycles

010: 13.5 cycles

011: 28.5 cycles

100: 41.5 cycles

101: 55.5 cycles

110: 71.5 cycles

111: 239.5 cycles

ADC1 analog Channel16 and Channel 17 are internally connected to the temperature sensor and to V_{REFINT}, respectively.

10.11.5 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
Res.		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
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:0 **SMPx[2:0]:** Channel x Sample time selection

These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged.

000: 1.5 cycles
001: 7.5 cycles
010: 13.5 cycles
011: 28.5 cycles
100: 41.5 cycles
101: 55.5 cycles
110: 71.5 cycles
111: 239.5 cycles

10.11.6 ADC injected channel data offset register x (ADC_JOFRx)(x=1..4)

Address offset: 0x14-0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				JOFFSETx[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **JOFFSETx[11:0]**: Data offset for injected channel x

These bits are written by software to define the offset to be subtracted from the raw converted data when converting injected channels. The conversion result can be read from in the ADC_JDRx registers.

10.11.7 ADC watchdog high threshold register (ADC_HTR)

Address offset: 0x24

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **HT[11:0]:** Analog watchdog high threshold

These bits are written by software to define the high threshold for the analog watchdog.

10.11.8 ADC watchdog low threshold register (ADC_LTR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LT[11:0]:** Analog watchdog low threshold

These bits are written by software to define the low threshold for the analog watchdog.

10.11.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:20 **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

Bits 19:15 **SQ16[4:0]**: 16th conversion in regular sequence

These bits are written by software with the channel number (0..17) assigned as the 16th in the conversion sequence.

Bits 14:10 **SQ15[4:0]**: 15th conversion in regular sequence

Bits 9:5 **SQ14[4:0]**: 14th conversion in regular sequence

Bits 4:0 **SQ13[4:0]**: 13th conversion in regular sequence

10.11.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		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
SQ10_0	SQ9[4:0]						SQ8[4:0]					SQ7[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:26 **SQ12[4:0]**: *1*2th conversion in regular sequence
These bits are written by software with the channel number (0..17) assigned as the 12th in the sequence to be converted.

Bits 24:20 **SQ11[4:0]**: 11th conversion in regular sequence

Bits 19:15 **SQ10[4:0]**: 10th conversion in regular sequence

Bits 14:10 **SQ9[4:0]:** 9th conversion in regular sequence

Bits 9:5 **SQ8[4:0]**: 8th conversion in regular sequence

Bits 4:0 **SQ7[4:0]**: 7th conversion in regular sequence

10.11.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		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
SQ4_0		SQ3[4:0]					SQ2[4:0]					SQ1[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:25 **SQ6[4:0]**: 6th conversion in regular sequence
These bits are written by software with the channel number (0..17) assigned as the 6th in the sequence to be converted.

Bits 24:20 **SQ5[4:0]:** 5th conversion in regular sequence

Bits 19:15 **SQ4[4:0]**: 4th conversion in regular sequence

Bits 14:10 **SQ3[4:0]**: 3rd conversion in regular sequence

Bits 9:5 **SQ2[4:0]:** 2nd conversion in regular sequence

Bits 4:0 **SQ1[4:0]**: 1st conversion in regular sequence

10.11.12 ADC injected sequence register (ADC_JSQR)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4_0	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **JL[1:0]**: Injected 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

Bits 19:15 **JSQ4[4:0]**: 4th conversion in injected sequence (when JL[1:0] = 3)(1)

These bits are written by software with the channel number (0..17) assigned as the 4th in the sequence to be converted.

Note: Unlike a regular conversion sequence, if JL[1:0] length is less than four, the channels are converted in a sequence starting from (4-JL). Example: ADC_JSQR[21:0] = 10 00011 00011 00111 00010 means that a scan conversion will convert the following channel sequence: 7, 3, 3. (not 2, 7, 3)

Bits 14:10 **JSQ3[4:0]:** 3rd conversion in injected sequence (when JL[1:0] = 3)

Bits 9:5 **JSQ2[4:0]**: 2nd conversion in injected sequence (when JL[1:0] = 3)

Bits 4:0 **JSQ1[4:0]**: 1st conversion in injected sequence (when JL[1:0] = 3)

1. When JL=3 (4 injected conversions in the sequencer), the ADC converts the channels in this order: JSQ1[4:0] >> JSQ2[4:0] >> JSQ3[4:0] >> JSQ4[4:0]
When JL=2 (3 injected conversions in the sequencer), the ADC converts the channels in this order: JSQ2[4:0] >> JSQ3[4:0] >> JSQ4[4:0]
When JL=1 (2 injected conversions in the sequencer), the ADC converts the channels in this order: JSQ3[4:0] >> JSQ4[4:0]
When JL=0 (1 injected conversion in the sequencer), the ADC converts only JSQ4[4:0] channel

10.11.13 ADC injected data register x (ADC_JDRx) (x= 1..4)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **JDATA[15:0]**: Injected data

These bits are read only. They contain the conversion result from injected channel x. The data is left or right-aligned as shown in *Figure 29* and *Figure 30*.

10.11.14 ADC regular data register (ADC_DR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **DATA[15:0]**: Regular data

These bits are read only. They contain the conversion result from the regular channels. The data is left or right-aligned as shown in *Figure 29* and *Figure 30*.

10.11.15 ADC register map

The following table summarizes the ADC registers.

Table 62. ADC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_SR**	Reserved																											STRT	JSTRT	JEOC	EOC	AWD
	Reset value																												0	0	0	0	0
0x04	**ADC_CR1**	Reserved								AWDEN	JAWDEN	Reserved						DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWD SGL	SCAN	JEOC IE	AWDIE	EOCIE	AWDCH[4:0]				
	Reset value									0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR2**	Reserved								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL [2:0]			Reserved	JEXTTRIG	JEXTSEL [2:0]			ALIGN	Reserved		DMA	Reserved			RSTCAL	CAL	CONT	ADON
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0			0				0	0	0	0
0x0C	**ADC_SMPR1**	Sample time bits SMPx_x																															
	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	**ADC_SMPR2**	Sample time bits SMPx_x																															
	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	**ADC_JOFR1**	Reserved																				JOFFSET1[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**ADC_JOFR2**	Reserved																				JOFFSET2[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**ADC_JOFR3**	Reserved																				JOFFSET3[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x20	**ADC_JOFR4**	Reserved																				JOFFSET4[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_HTR**	Reserved																				HT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x28	**ADC_LTR**	Reserved																				LT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**ADC_SQR1**	Reserved								L[3:0]				SQ16[4:0] 16th conversion in regular sequence bits					SQ15[4:0] 15th conversion in regular sequence bits					SQ14[4:0] 14th conversion in regular sequence bits					SQ13[4:0] 13th conversion in regular sequence bits				
	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
0x30	**ADC_SQR2**	Reserved		SQ12[4:0] 12th conversion in regular sequence bits						SQ11[4:0] 11th conversion in regular sequence bits				SQ10[4:0] 10th conversion in regular sequence bits					SQ9[4:0] 9th conversion in regular sequence bits					SQ8[4:0] 8th conversion in regular sequence bits					SQ7[4:0] 7th conversion in regular sequence bits				
	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
0x34	**ADC_SQR3**	Reserved		SQ6[4:0] 6th conversion in regular sequence bits						SQ5[4:0] 5th conversion in regular sequence bits				SQ4[4:0] 4th conversion in regular sequence bits					SQ3[4:0] 3rd conversion in regular sequence bits					SQ2[4:0] 2nd conversion in regular sequence bits					SQ1[4:0] 1st conversion in regular sequence bits				
	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

Table 62. ADC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38	**ADC_JSQR**	Reserved										JL[1:0]		JSQ4[4:0] 4th conversion in injected sequence bits					JSQ3[4:0] 3rd conversion in injected sequence bits					JSQ2[4:0] 2nd conversion in injected sequence bits					JSQ1[4:0] 1st conversion in injected sequence bits				
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**ADC_JDR1**	Reserved																	JDATA[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**ADC_JDR2**	Reserved																	JDATA[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**ADC_JDR3**	Reserved																	JDATA[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**ADC_JDR4**	Reserved																	JDATA[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**ADC_DR**	Reserved																	Regular DATA[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

11 Digital-to-analog converter (DAC)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

11.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC has two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, V_{REF+} (shared with ADC) is available for better resolution.

11.2 DAC main features

- Two DAC converters: one output channel each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel
- DMA underrun error detection
- External triggers for conversion
- Input voltage reference, V_{REF+}

Figure 32 shows the block diagram of a DAC channel and *Table 63* gives the pin description.

Figure 32. DAC channel block diagram

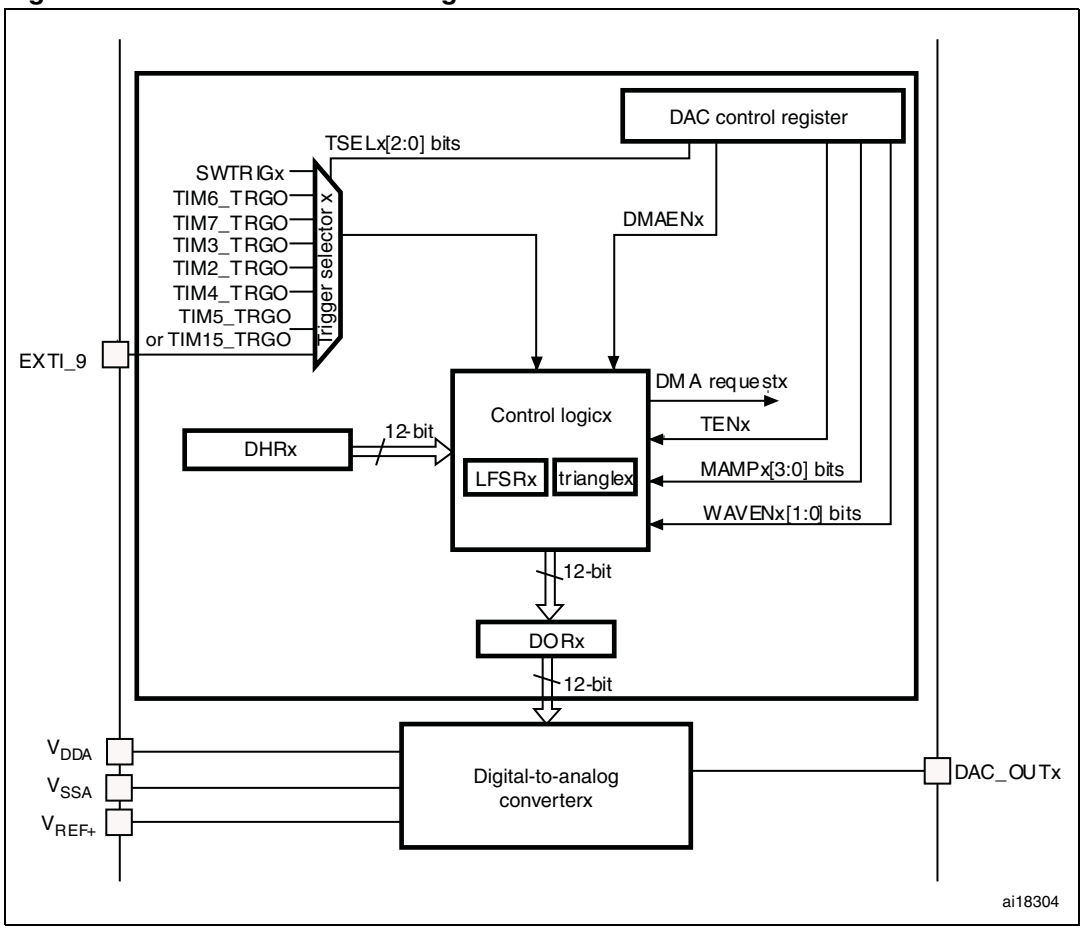


Table 63. DAC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the DAC, 2.4 V ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}	Input, analog supply	Analog power supply
V_{SSA}	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Note: Once the DAC channelx is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DAC_OUTx). In order to avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN).

11.3 DAC functional description

11.3.1 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a startup time t_{WAKEUP}.

Note: The ENx bit enables the analog DAC Channelx macrocell only. The DAC Channelx digital interface is enabled even if the ENx bit is reset.

11.3.2 DAC output buffer enable

The DAC integrates two output buffers that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. Each DAC channel output buffer can be enabled and disabled using the corresponding BOFFx bit in the DAC_CR register.

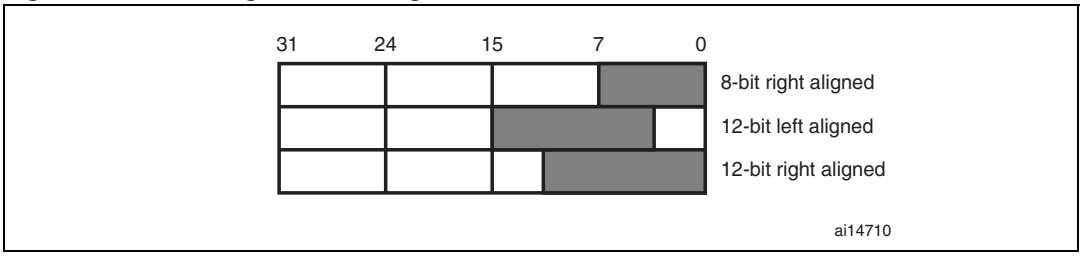
11.3.3 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channelx, there are three possibilities:
  - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx [7:0] bits (stored into the DHRx[11:4] bits)
  - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
  - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

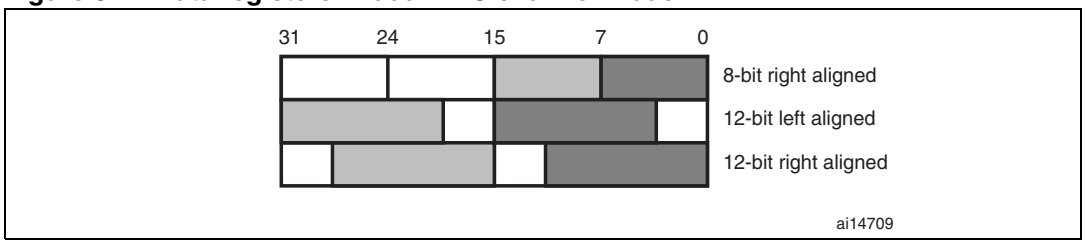
Figure 33. Data registers in single DAC channel mode



- Dual DAC channels, there are three possibilities:
  - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
  - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 34. Data registers in dual DAC channel mode



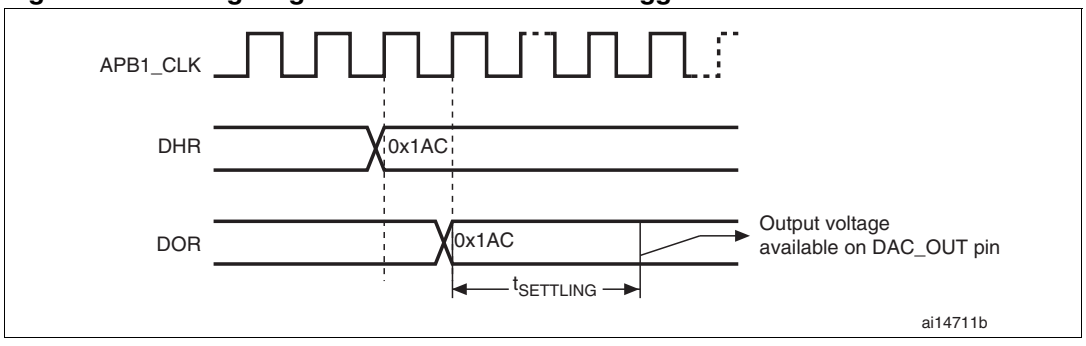
11.3.4 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three APB1 clock cycles later.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 35. Timing diagram for conversion with trigger disabled TEN = 0



11.3.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4095}$$

11.3.6 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which out of 8 possible events will trigger conversion as shown in *Table 64*.

Table 64. External triggers

Source	Type	TSEL[2:0]
Timer 6 TRGO event	Internal signal from on-chip timers	000
Timer 3 TRGO event		001
Timer 7 TRGO event		010
Timer 5 or Timer 15 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the selected external interrupt line 9, the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: 1 TSELx[2:0] bit cannot be changed when the ENx bit is set.

2 When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one APB1 clock cycle.

3 For High-density value line devices, the TIM15 TRGO event is selected as DAC Trigger 3 only if the MISC_REMAP bit in the AFIO_MAPR2 register is set and DAC Trigger 3 is connected to TIM5 TRGO when the MISC_REMAP bit in the AFIO_MAPR2 register is reset. On low- and medium -density devices the TIM15 TRGO event is always selected as DAC Trigger 3. For more details please refer to the Section 7.3: Alternate function I/O and debug configuration (AFIO) on page 114.

11.3.7 DMA request

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the DMAENx bit is set. The value of the DAC_DHRx register is then transferred into the DAC_DORx register.
In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, you should set only the corresponding DMAENx bit. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgement for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. DMA data transfers are then disabled and no further DMA request is treated. The DAC channelx continues to convert old data.

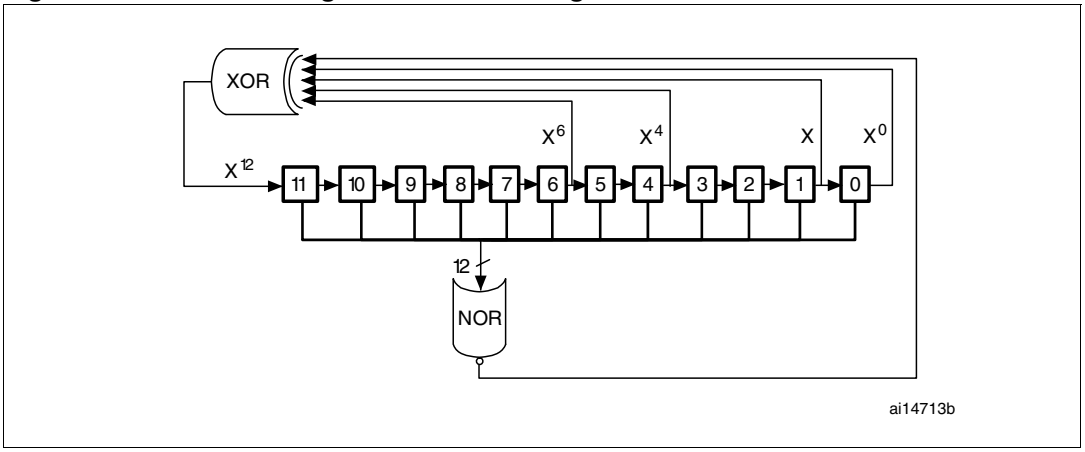
The software should clear the DMAUDRx flag by writing “1”, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channlex, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

11.3.8 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to “01”. The preloaded value in LFSR is 0xAAA. This register is updated three APB1 clock cycles after each trigger event, following a specific calculation algorithm.

Figure 36. DAC LFSR register calculation algorithm

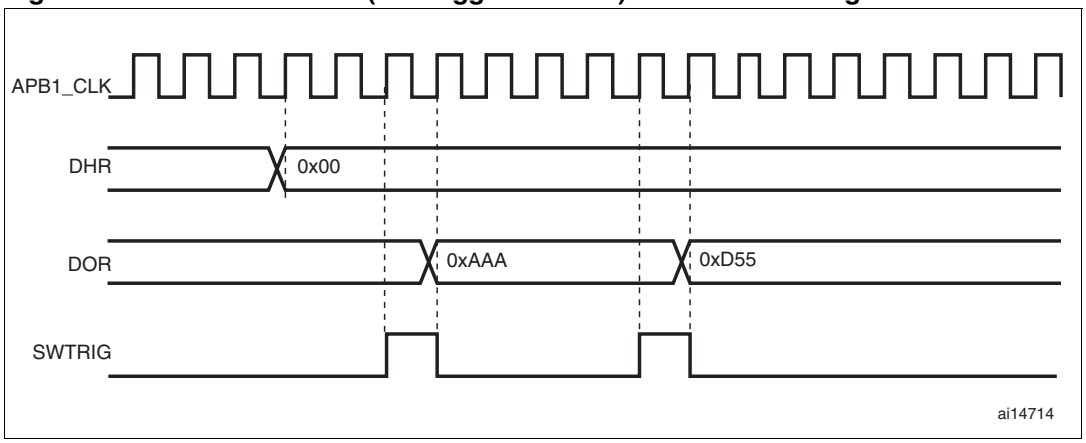


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then stored into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 37. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

11.3.9 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to “10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB1 clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is stored into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 38. DAC triangle wave generation

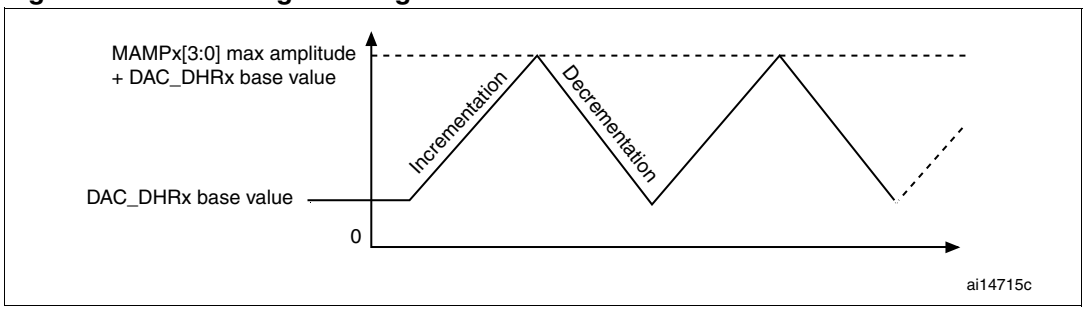
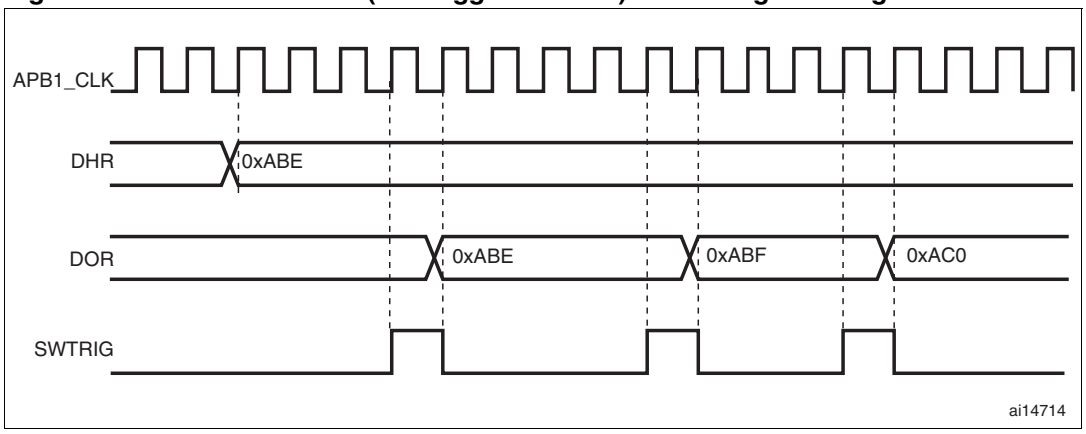


Figure 39. DAC conversion (SW trigger enabled) with triangle wave generation



Note: 1 The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

2 The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

11.4 Dual DAC channel conversion

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time.

Eleven possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

11.4.1 Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three APB1 clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three APB1 clock cycles later).

11.4.2 Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

11.4.3 Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

11.4.4 Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into

DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

11.4.5 Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

11.4.6 Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

In this configuration, one APB1 clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

11.4.7 Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three APB1 clock cycles).

11.4.8 Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

11.4.9 Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

11.4.10 Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value using the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.
At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is

added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

11.4.11 Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the DAC channel1 triangle counter is updated.
At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the DAC channel2 triangle counter is updated.

11.5 DAC registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

11.5.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		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
Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved.

Bits 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable
This bit is set and cleared by software.
0: DAC channel2 DMA underrun interrupt disabled
1: DAC channel2 DMA underrun interrupt enabled

Bit 28 **DMAEN2**: DAC channel2 DMA enable
This bit is set and cleared by software.
0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled

Bit 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bit 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

Bits 21:19 **TSEL2[2:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

000: Timer 6 TRGO event
001: Timer 3 TRGO event
010: Timer 7 TRGO event
011: Timer 5 or timer 15 TRGO event
100: Timer 2 TRGO event
101: Timer 4 TRGO event
110: External line9
111: Software trigger

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
For high-density value line devices, the TIM15 TRGO event is selected as DAC Trigger 3 only if the MISC_REMAP bit in the AFIO_MAPR2 register is set and DAC Trigger 3 is connected to TIM5 TRGO when the MISC_REMAP bit in the AFIO_MAPR2 register is reset. On low- and medium -density devices the TIM15 TRGO event is always selected as DAC Trigger 3. For more details please refer to the AFIO section.

Bit 18 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR2 register

*Note: When software trigger is selected, the transfer from the DAC_DHRx reg*ister to the DAC_DOR2 register takes only one APB1 clock cycle.*

Bit 17 **BOFF2**: DAC channel2 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel2 output buffer.

0: DAC channel2 output buffer enabled
1: DAC channel2 output buffer disabled

Bit 16 **EN2**: DAC channel2 enable
This bit is set and cleared by software to enable/disable DAC channel2.
0: DAC channel2 disabled
1: DAC channel2 enabled

Bits 15:14 Reserved.

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable
This bit is set and cleared by software.
0: DAC channel1 DMA Underrun Interrupt disabled
1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable
This bit is set and cleared by software.
0: DAC channel1 DMA mode disabled
1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable
These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled
Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:3 **TSEL1[2:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1.
000: Timer 6 TRGO event
001: Timer 3 TRGO event
010: Timer 7 TRGO event
011: Timer 5 or Timer 15 TRGO event
100: Timer 2 TRGO event
101: Timer 4 TRGO event
110: External line9
111: Software trigger
Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 2 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.

0: DAC channel1 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR1 register

1: DAC channel1 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DOR1 register takes only one APB1 clock cycle.

Bit 1 **BOFF1**: DAC channel1 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel1 output buffer.

0: DAC channel1 output buffer enabled

1: DAC channel1 output buffer disabled

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.

0: DAC channel1 disabled

1: DAC channel1 enabled

11.5.2 DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

11.5.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved.

Bit 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

11.5.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved.

Bit 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved.

11.5.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

11.5.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

11.5.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved.

11.5.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

11.5.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DACC2DHR[11:0]											
				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
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

11.5.10 DUAL DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Reserved			
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
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved.

11.5.11 DUAL DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

11.5.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved.

Bit 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

11.5.13 DAC channel2 data output register (DAC_DOR2)

Address offset: 0x30
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved.

Bit 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

11.5.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDR2	Reserved												
		rc_w1													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DMAUDR1	Reserved												
		rc_w1													

Bits 31:30 Reserved.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel2
1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate)

Bits 28:14 Reserved.

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel1
1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bits 12:0 Reserved.

11.5.15 DAC register map

Table 65 summarizes the DAC registers.

Table 65. DAC register map

Address offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE 2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE 1[2:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
0x04	DAC_SWTRIGR	Reserved																														SWTRIG2	SWTRIG1
0x08	DAC_DHR12R1	Reserved																DACC1DHR[11:0]															
0x0C	DAC_DHR12L1	Reserved																DACC1DHR[11:0]												Reserved			
0x10	DAC_DHR8R1	Reserved																								DACC1DHR[7:0]							
0x14	DAC_DHR12R2	Reserved																DACC2DHR[11:0]															
0x18	DAC_DHR12L2	Reserved																DACC2DHR[11:0]												Reserved			
0x1C	DAC_DHR8R2	Reserved																								DACC2DHR[7:0]							
0x20	DAC_DHR12RD	Reserved				DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]											
0x24	DAC_DHR12LD	DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]												Reserved			
0x28	DAC_DHR8RD	Reserved																DACC2DHR[7:0]								DACC1DHR[7:0]							
0x2C	DAC_DOR1	Reserved																				DACC1DOR[11:0]											
0x30	DAC_DOR2	Reserved																				DACC2DOR[11:0]											
0x34	DAC_SR	Reserved		DMAUDR2		Reserved														DMAUDR1		Reserved											

Refer to *Table 1: Low and medium-density device register boundary addresses on page 36* and *Table 2: High-density device register boundary addresses on page 37* for the register boundary addresses.

12 Advanced-control timers (TIM1)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

12.1 TIM1 introduction

The advanced-control timers (TIM1) consist 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 (TIMx) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 12.3.20*.

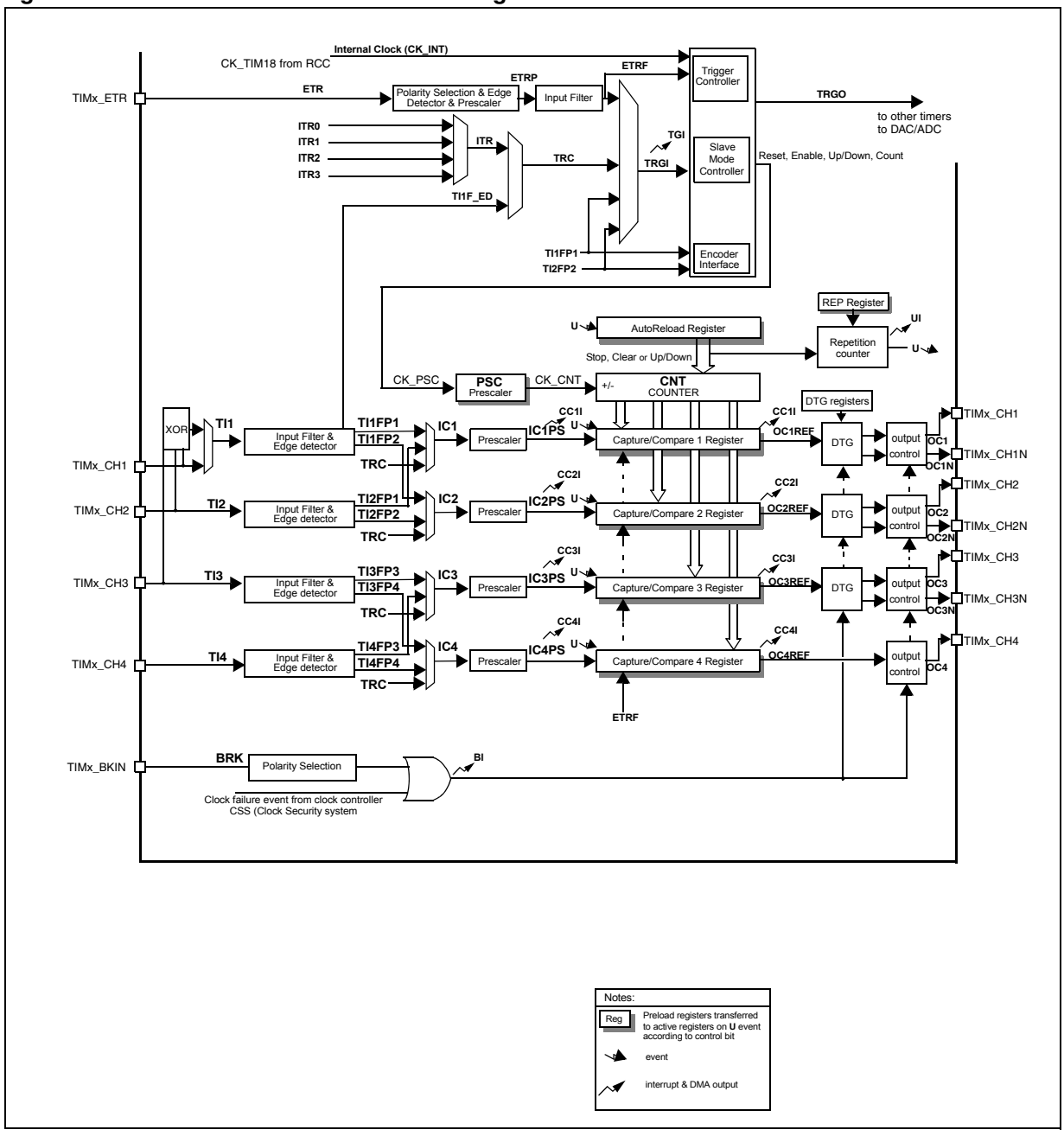
12.2 TIM1 main features

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 65535.
- Up to 4 independent channels for:
  - Input Capture
  - 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.
- Break input to put the timer’s output signals in reset state or in a known state.

- 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
  - Break input
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 40. Advanced-control timer block diagram



12.3 TIM1 functional description

12.3.1 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 42 and *Figure 43* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 41. Counter timing diagram with prescaler division change from 1 to 2

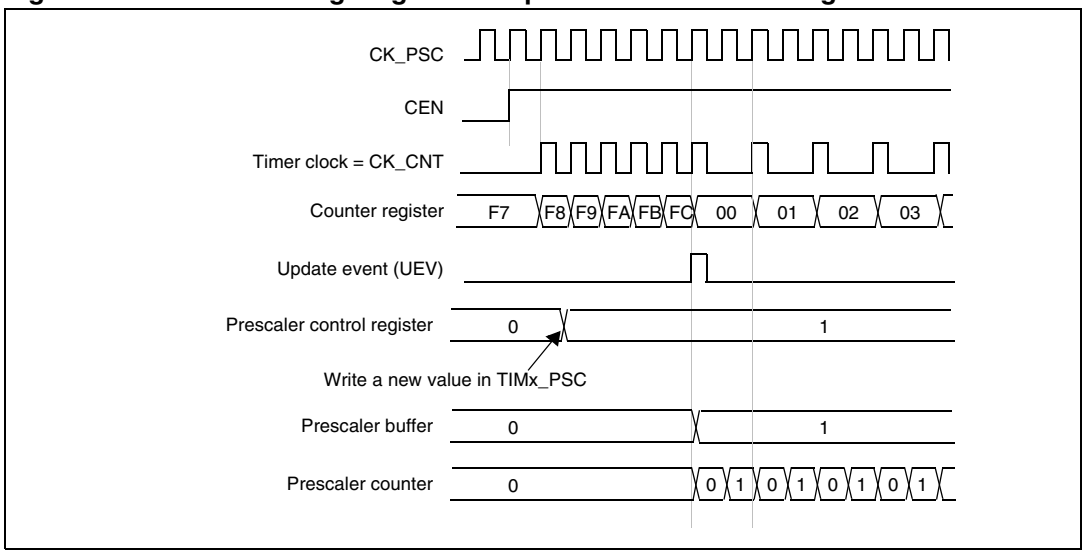
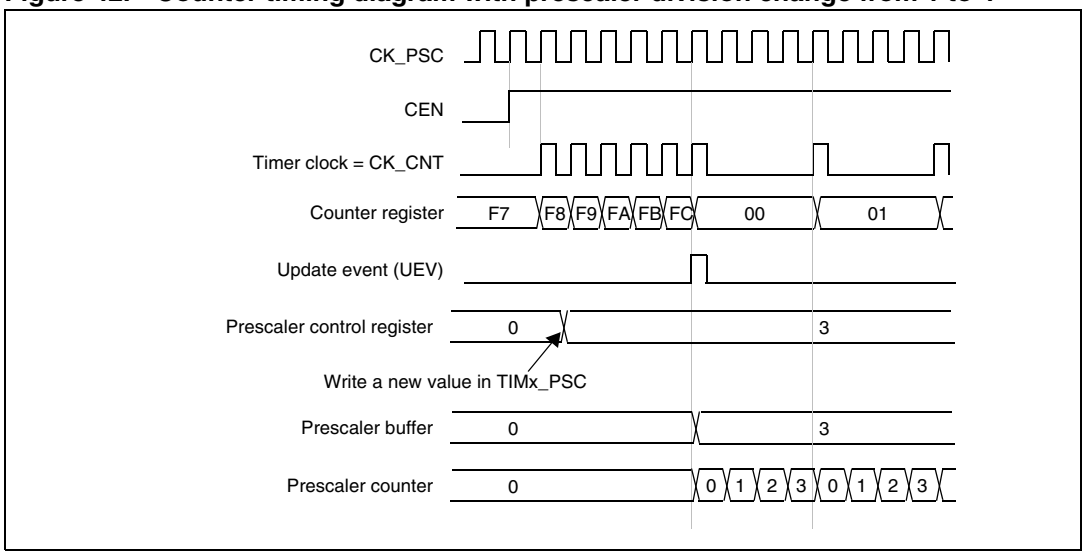


Figure 42. Counter timing diagram with prescaler division change from 1 to 4



12.3.2 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 43. Counter timing diagram, internal clock divided by 1

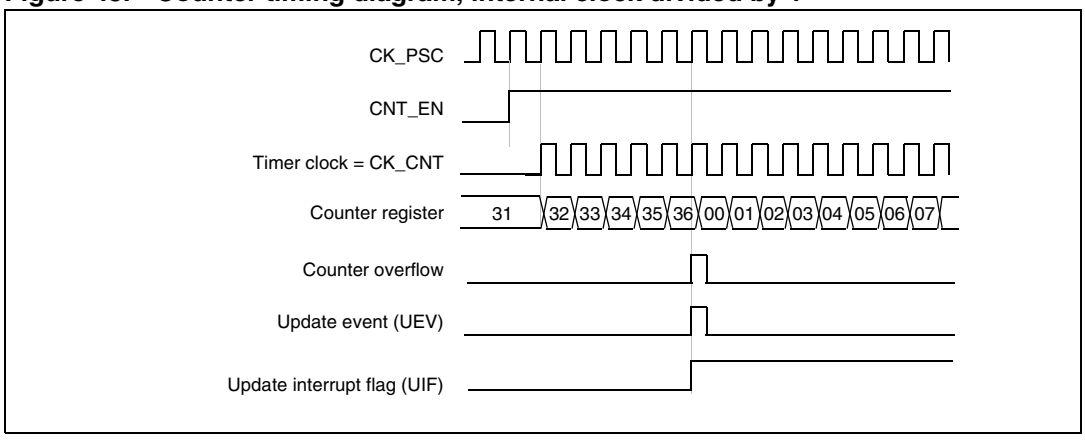


Figure 44. Counter timing diagram, internal clock divided by 2

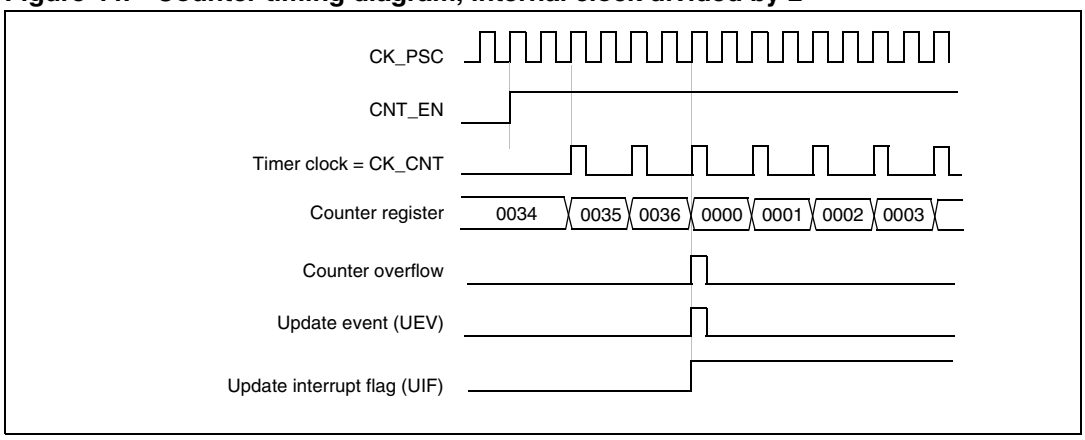


Figure 45. Counter timing diagram, internal clock divided by 4

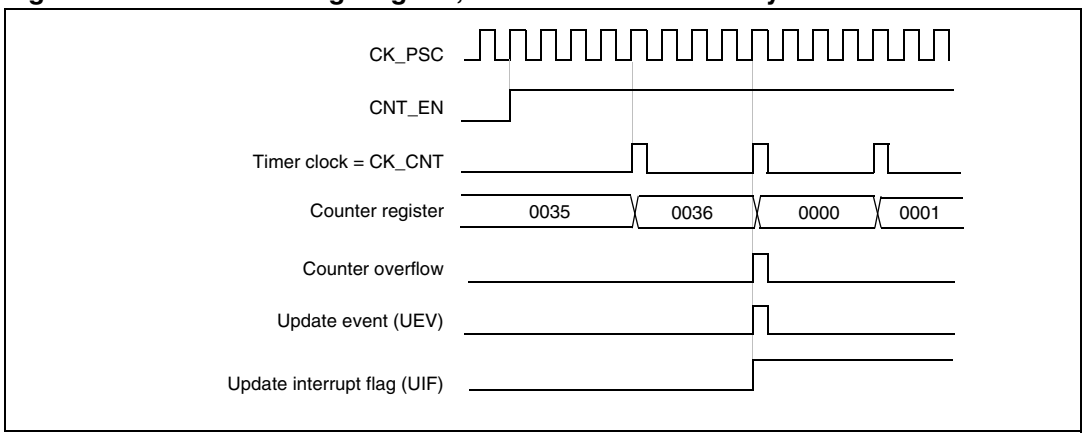


Figure 46. Counter timing diagram, internal clock divided by N

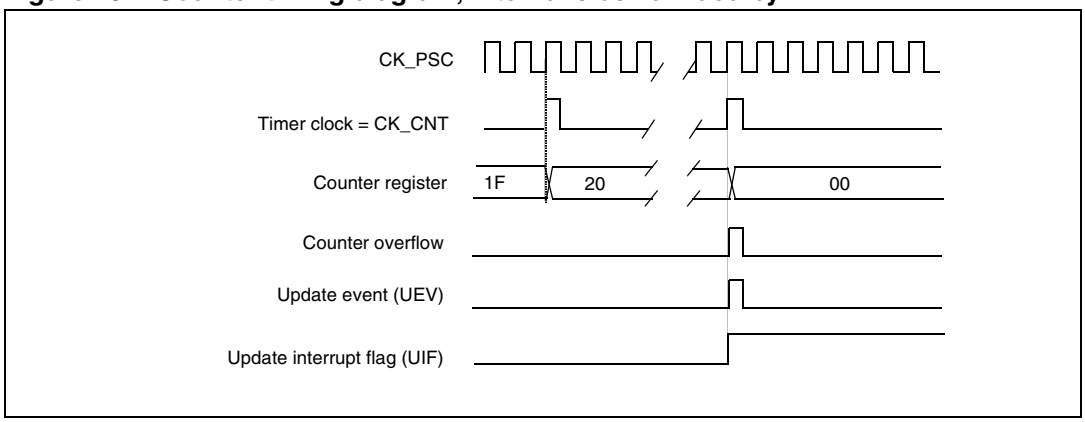


Figure 47. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

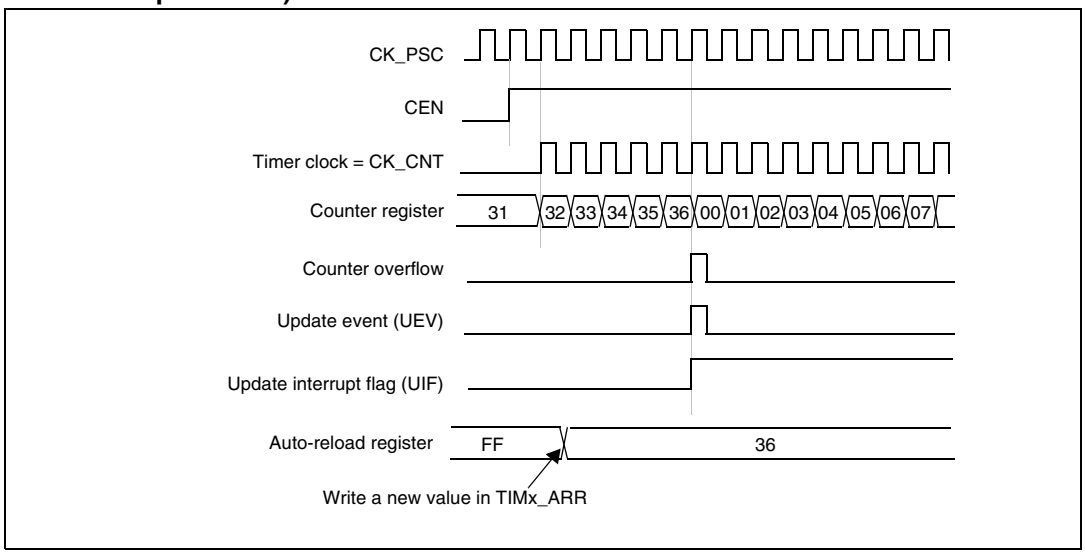
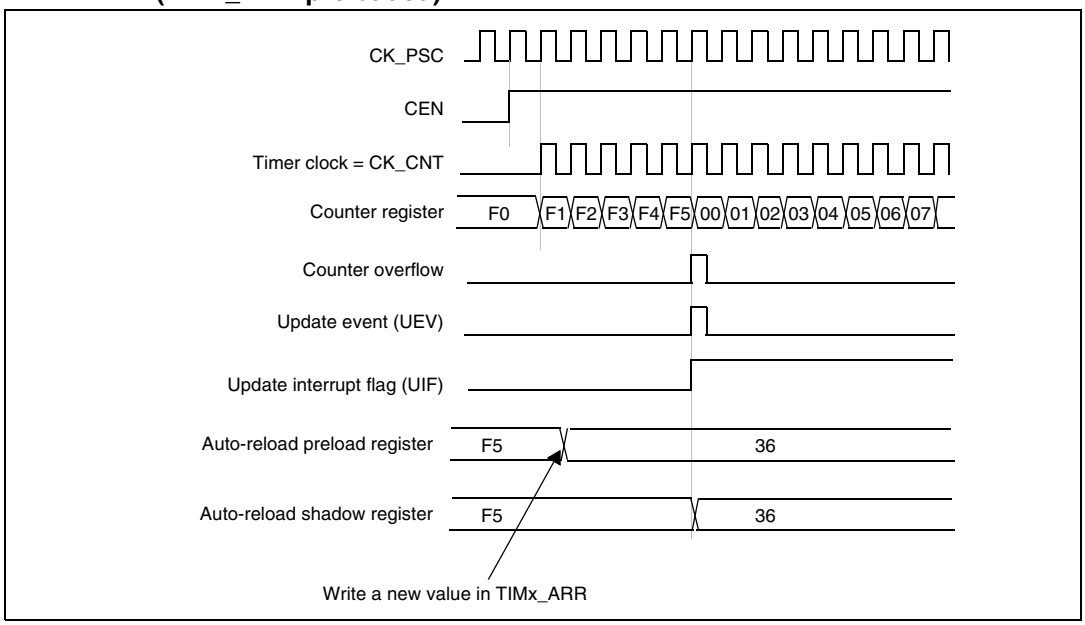


Figure 48. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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). 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 49. Counter timing diagram, internal clock divided by 1

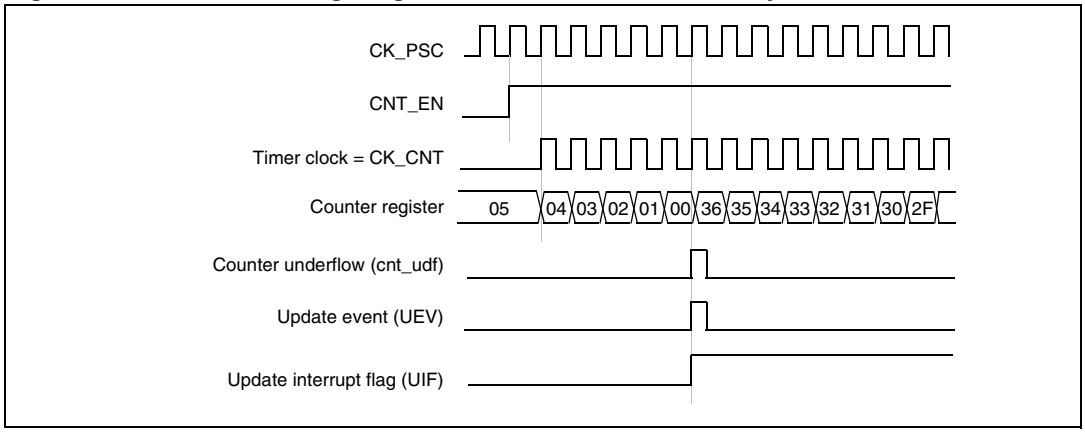


Figure 50. Counter timing diagram, internal clock divided by 2

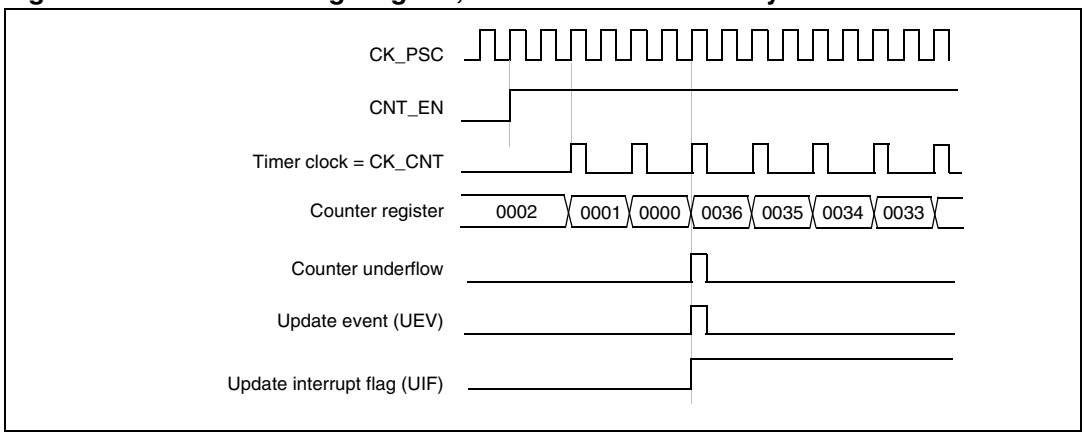


Figure 51. Counter timing diagram, internal clock divided by 4

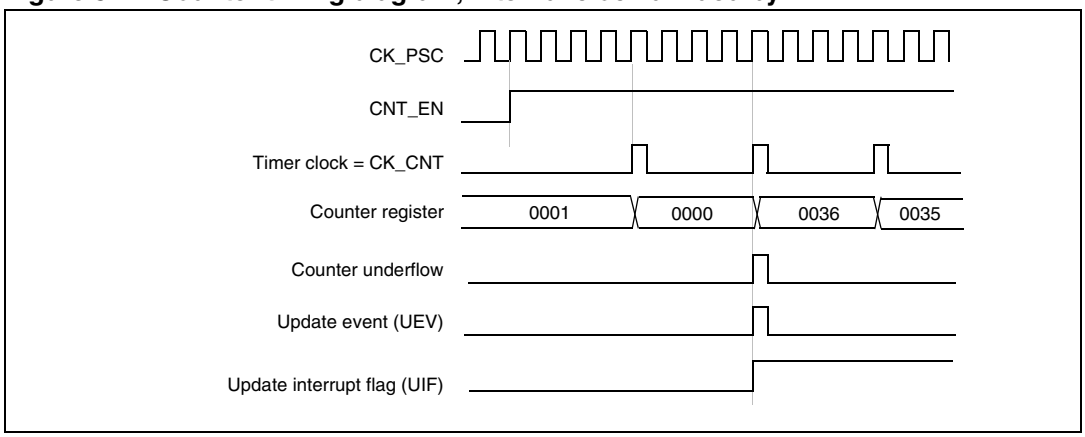


Figure 52. Counter timing diagram, internal clock divided by N

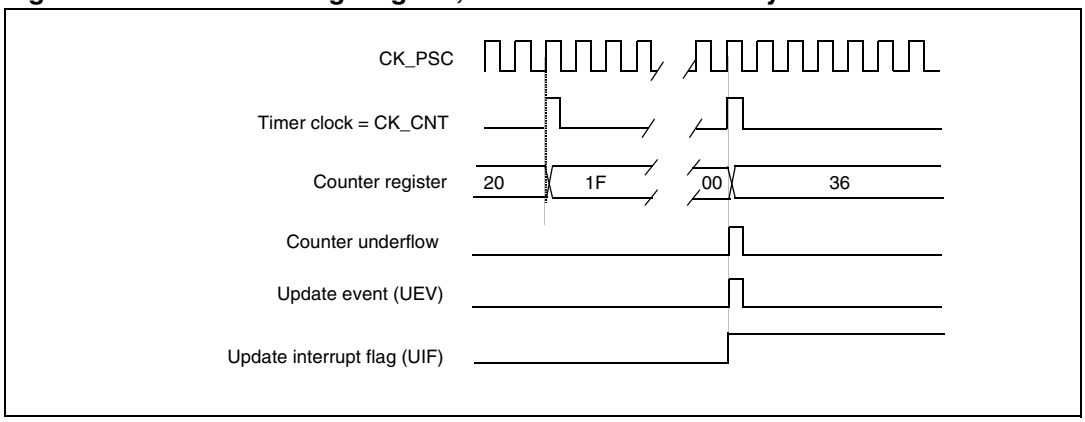
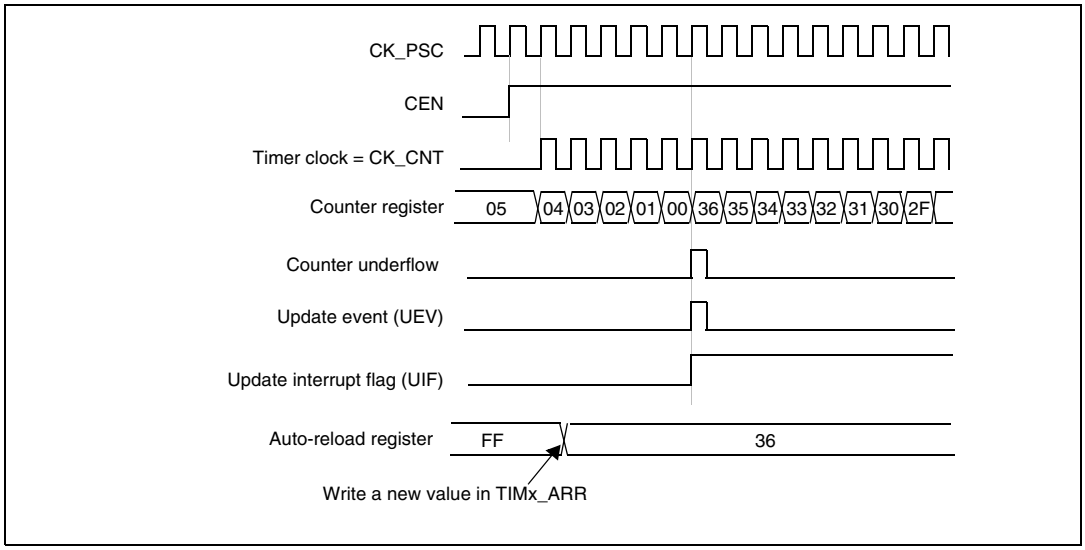


Figure 53. Counter timing diagram, update event when repetition counter is not used



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.

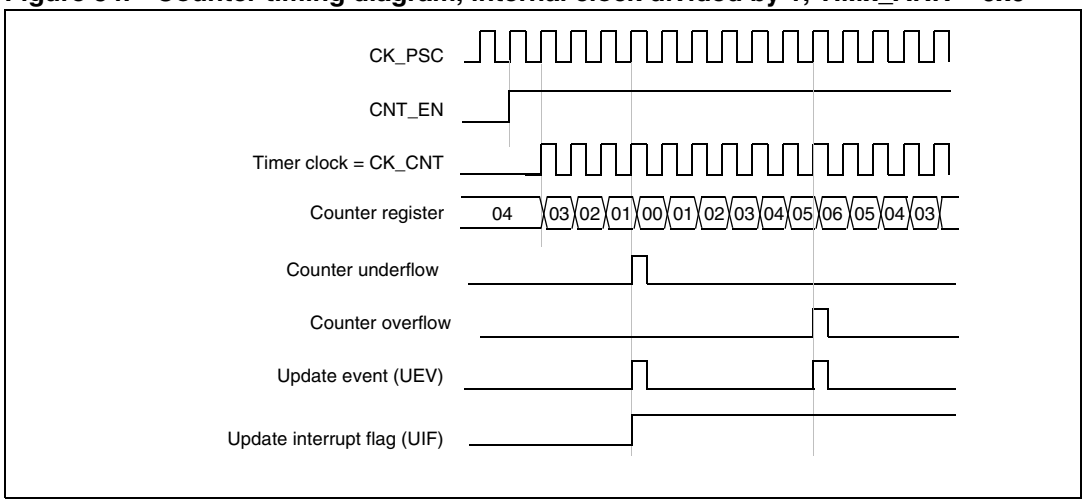
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 54. 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 12.4: TIM1 registers on page 248*).

Figure 55. Counter timing diagram, internal clock divided by 2

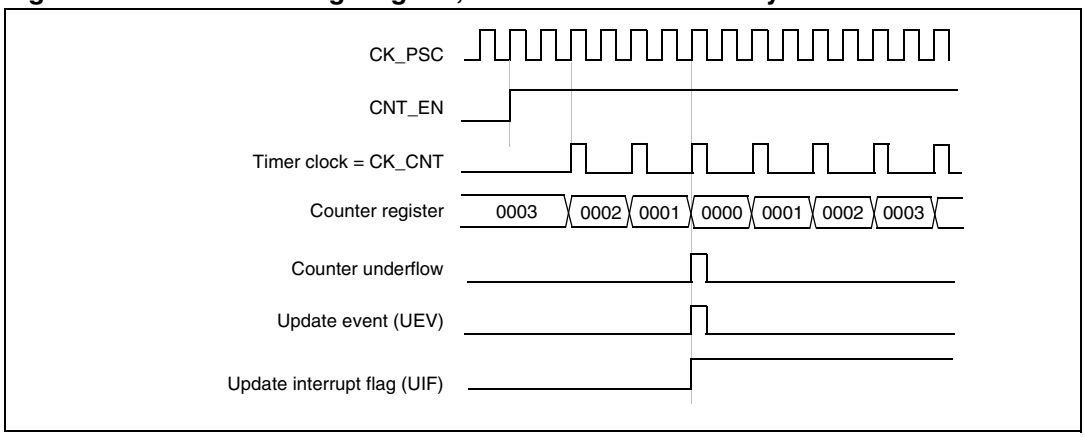
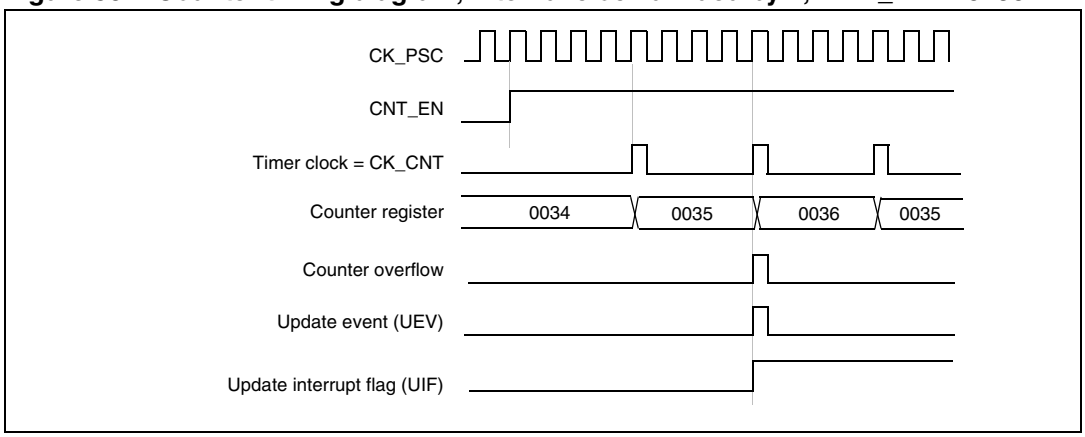


Figure 56. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 57. Counter timing diagram, internal clock divided by N

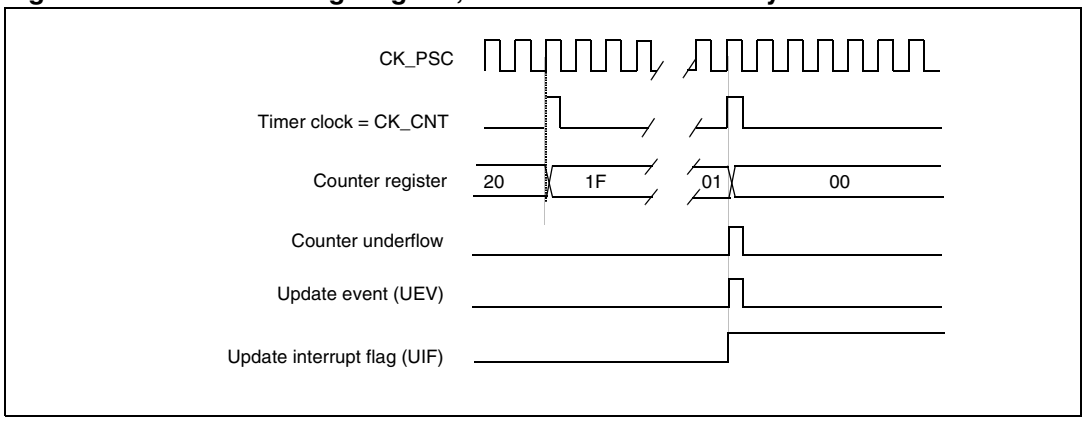


Figure 58. Counter timing diagram, update event with ARPE=1 (counter underflow)

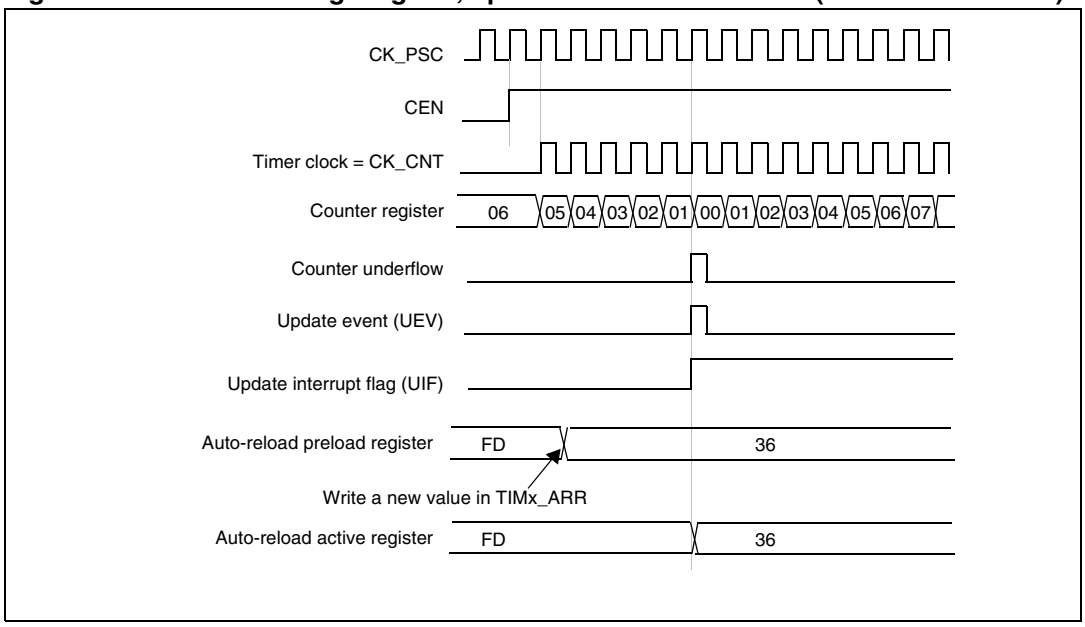
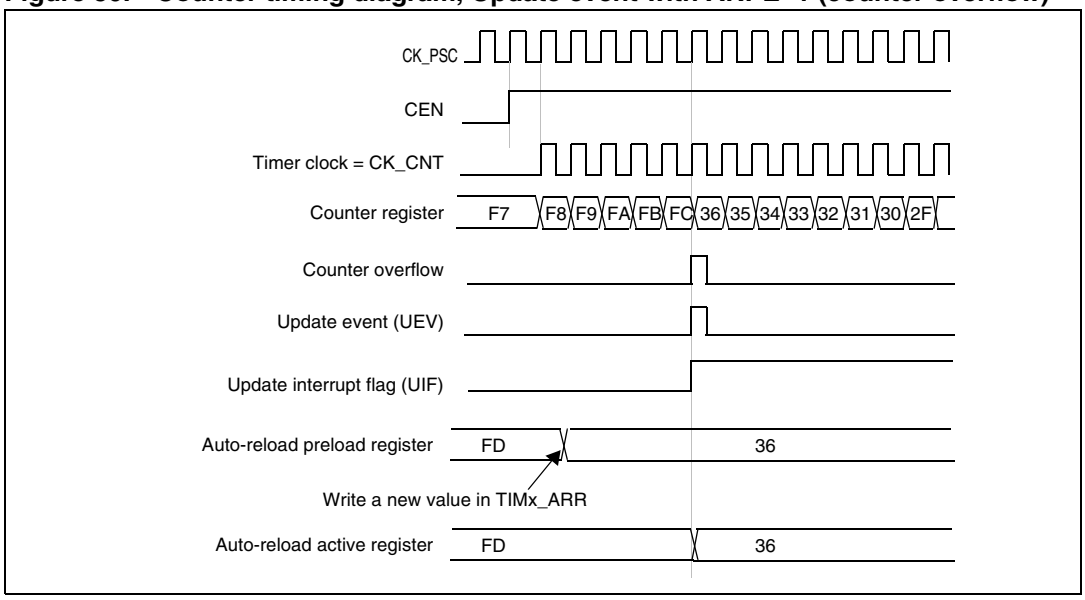


Figure 59. Counter timing diagram, Update event with ARPE=1 (counter overflow)



12.3.3 Repetition counter

Section 12.3.1: 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 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

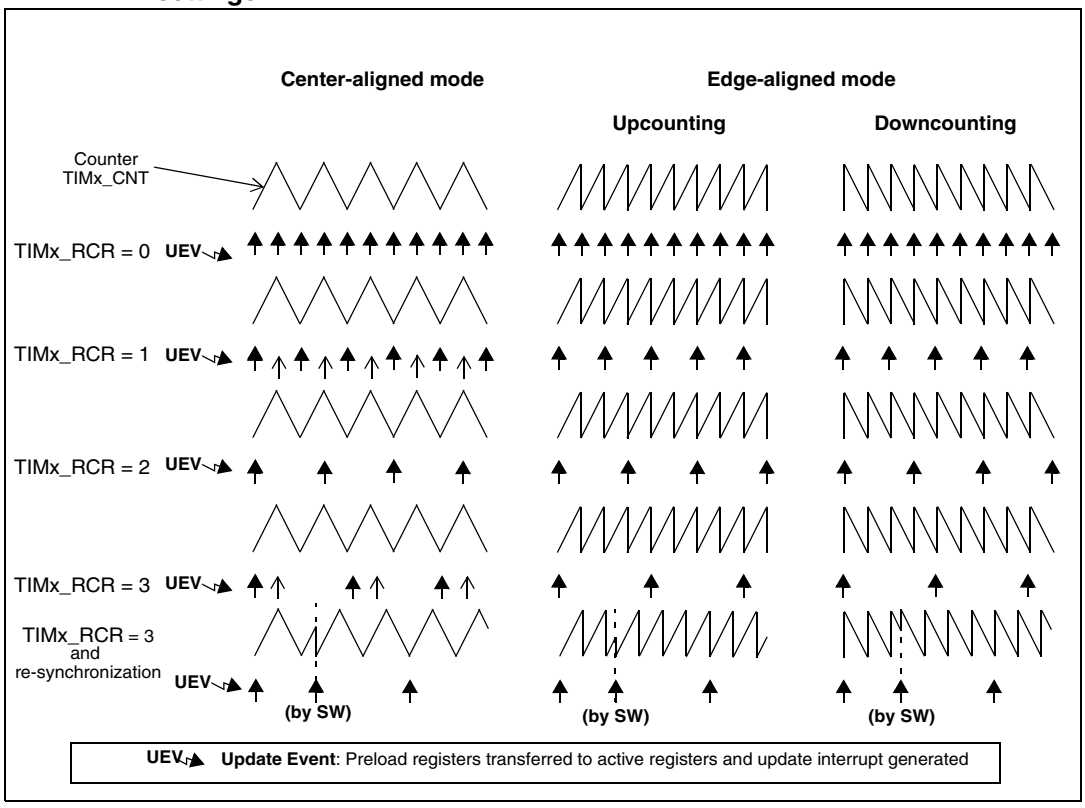
The repetition counter is decremented:

- 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 128 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 $2xT_{ck}$, 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 60*). 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 started. If the RCR was written before starting the counter, the UEV occurs on the overflow. If the RCR was written after starting the counter, the UEV occurs on the underflow. For example for RCR = 3, the UEV is generated on each 4th overflow or underflow event depending on when RCR was written.

Figure 60. Update rate examples depending on mode and TIMx_RCR register settings



12.3.4 Clock selection

The counter clock can be provided by the following clock sources:

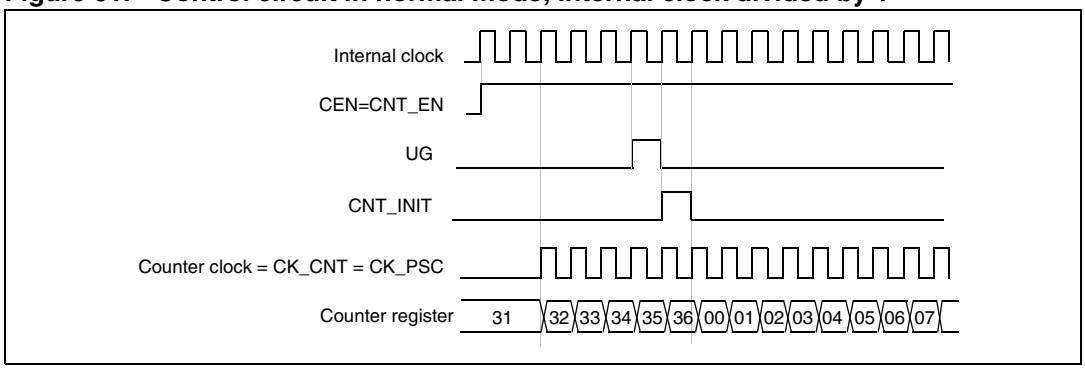
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer 1 to act as a prescaler for Timer 2. Refer to *Using one timer as prescaler for another* for more details.

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 61 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

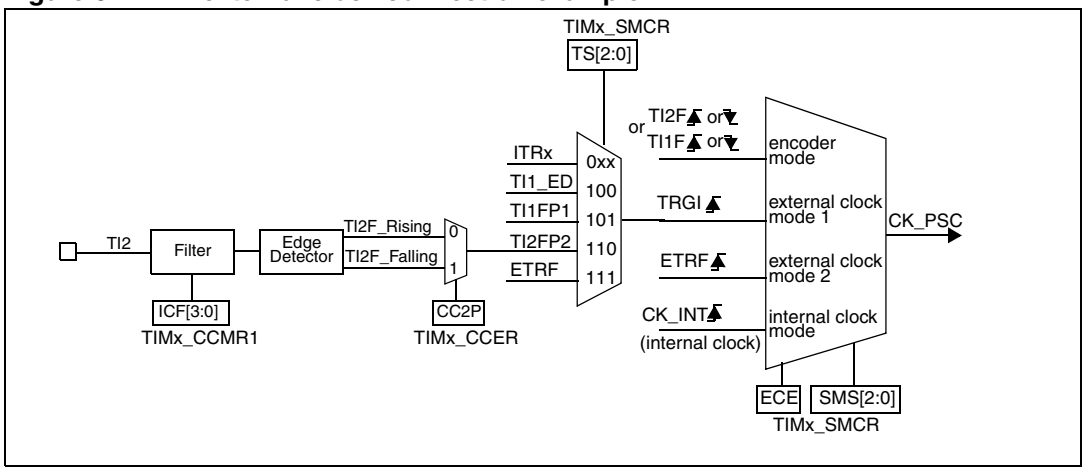
Figure 61. Control circuit in normal mode, internal clock divided by 1



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.

Figure 62. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

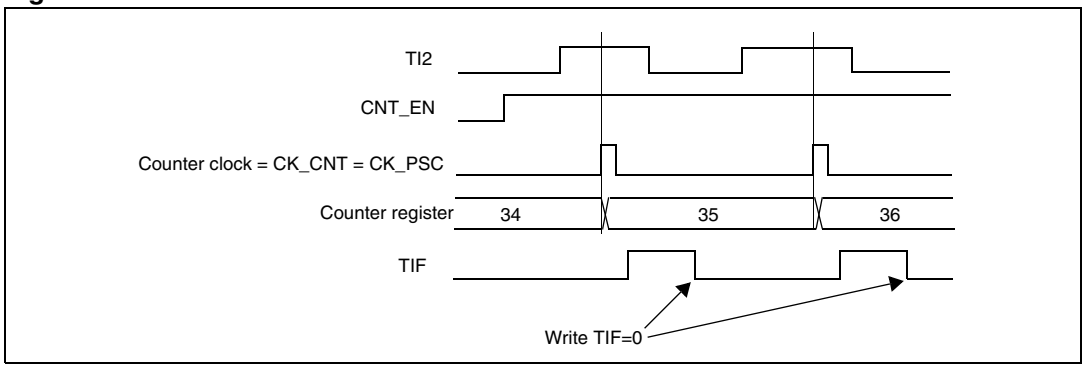
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don't 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.

Figure 63. Control circuit in external clock mode 1



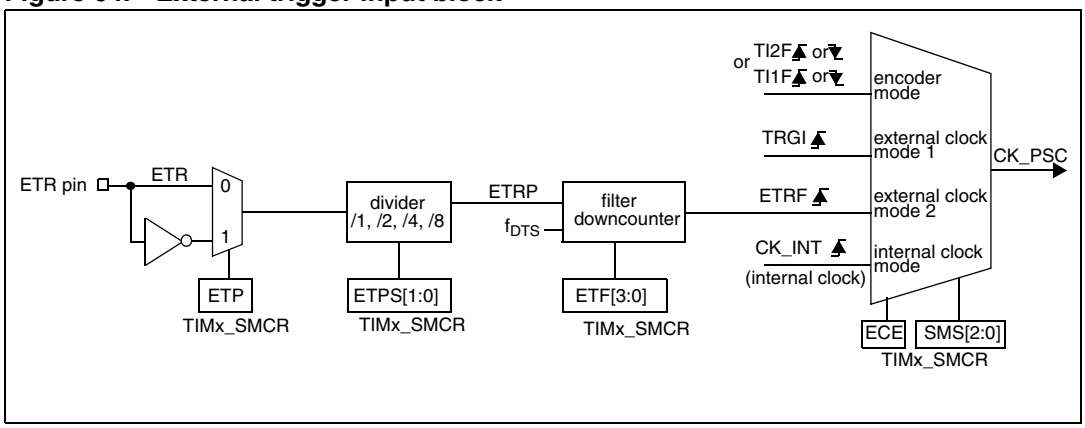
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 64* gives an overview of the external trigger input block.

Figure 64. External trigger input block



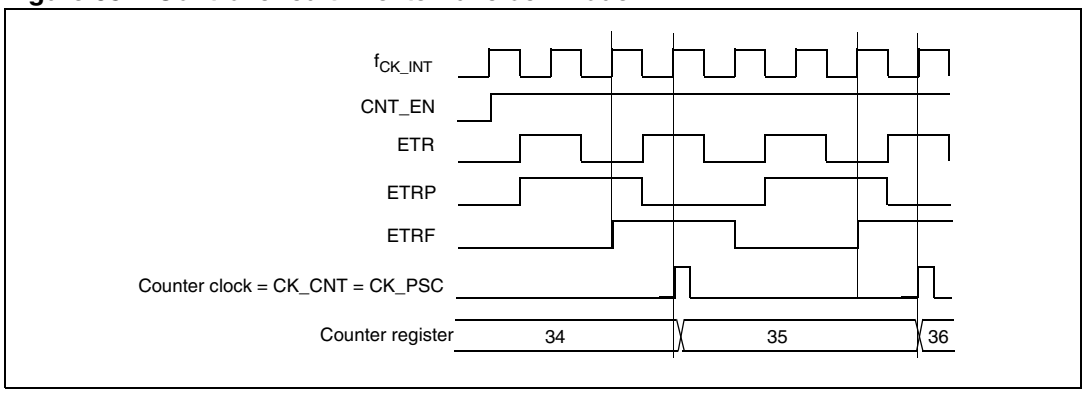
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 65. Control circuit in external clock mode 2



12.3.5 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 66 to *Figure 69* 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 66. 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 67. Capture/compare channel 1 main circuit

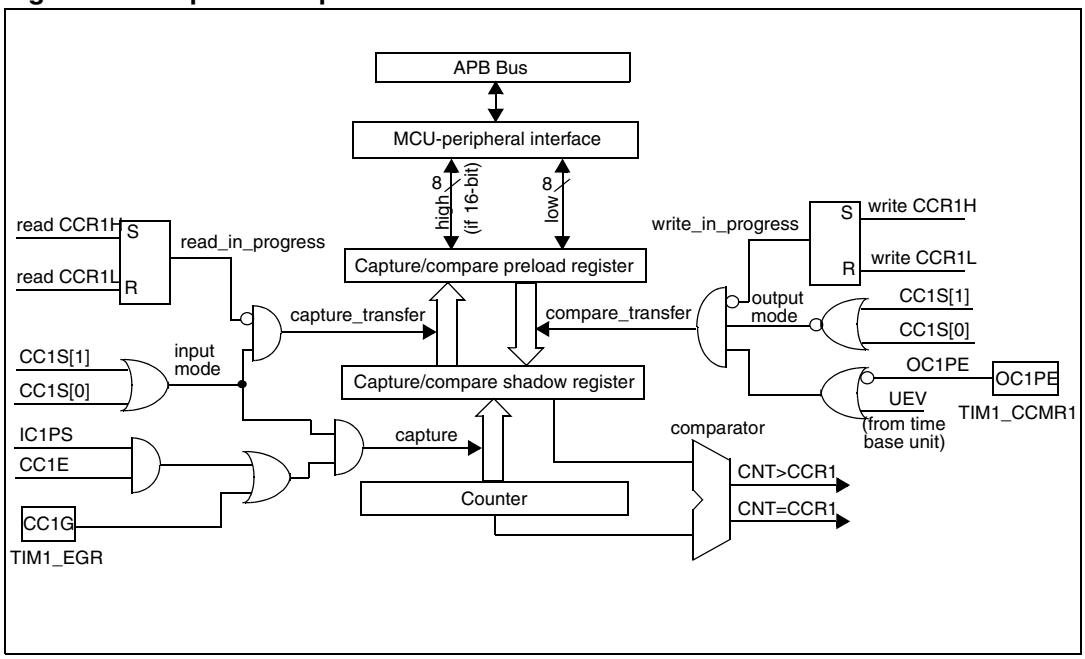


Figure 68. Output stage of capture/compare channel (channel 1 to 3)

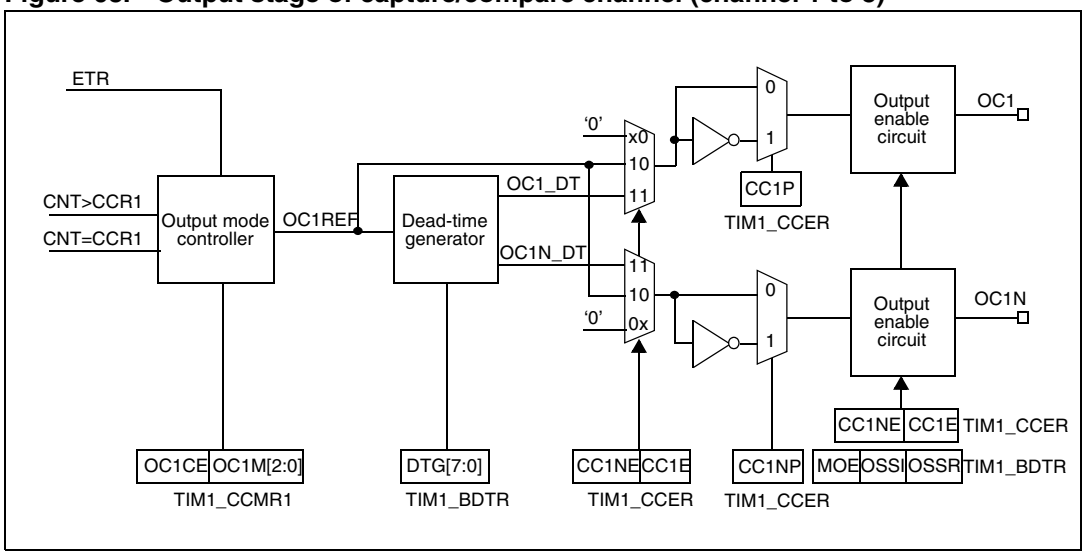
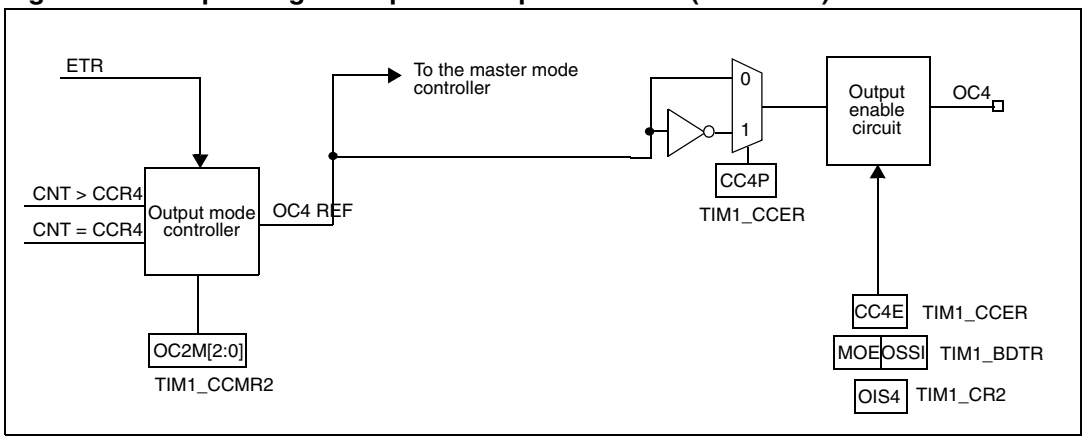


Figure 69. Output stage of capture/compare channel (channel 4)



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.

12.3.6 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 you write it to ‘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:

- 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.
- Program the input filter duration you need with respect to the signal you connect 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 f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- 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).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- 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.

12.3.7 PWM input mode

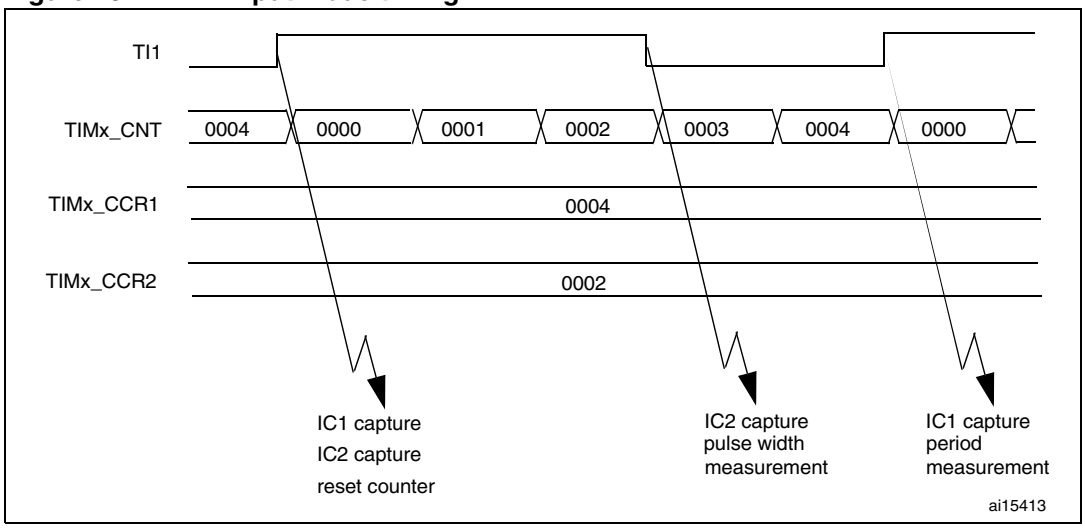
This mode is a particular case of input capture mode. The procedure is the same except:

- 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, you 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):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P bit to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 70. PWM input mode timing



12.3.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need 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.

12.3.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

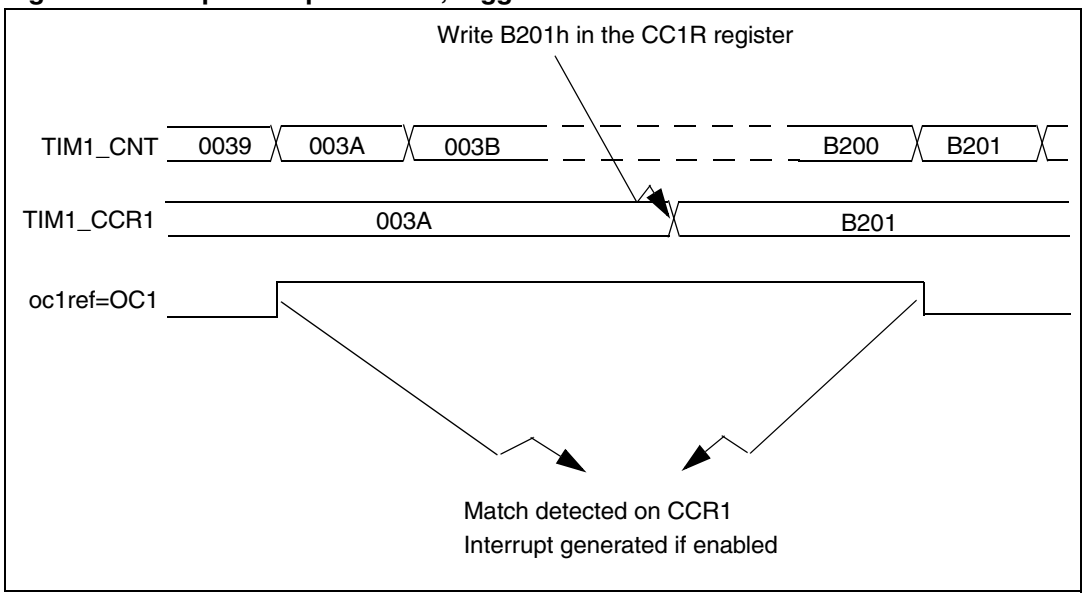
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 71*.

Figure 71. Output compare mode, toggle on OC1.



12.3.10 PWM mode

Pulse Width Modulation mode allows you 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. You must enable the corresponding preload register 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, you have to initialize all the registers 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.

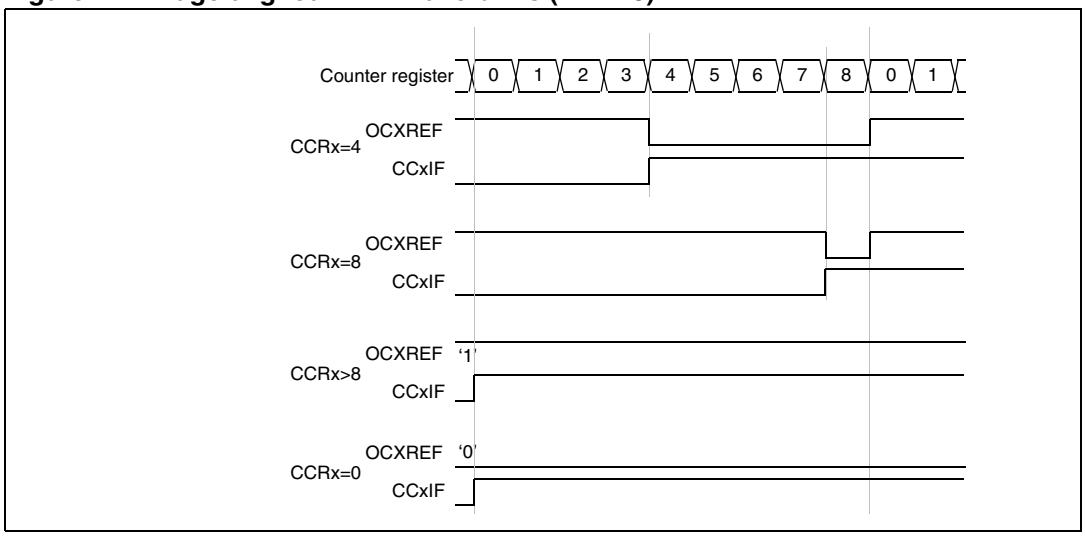
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 211*.

  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 72* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 72. 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 214*

  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.

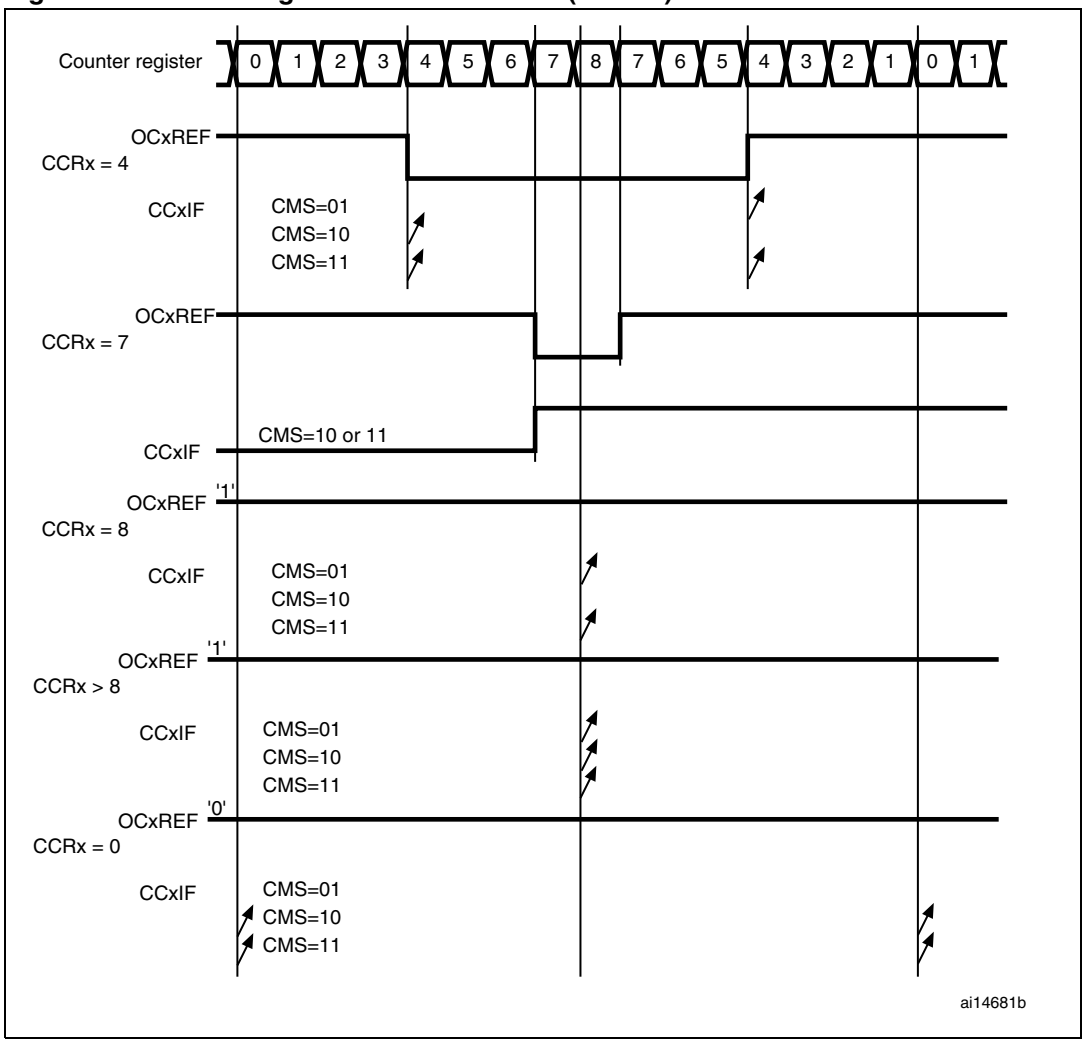
PWM center-aligned mode

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 216*.

Figure 73 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 73. Center-aligned PWM waveforms (ARR=8)



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 you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value 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.

12.3.11 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 you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) 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 68: Output control bits for complementary OCx and OCxN channels with break feature on page 264* 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 74. Complementary output with dead-time insertion.

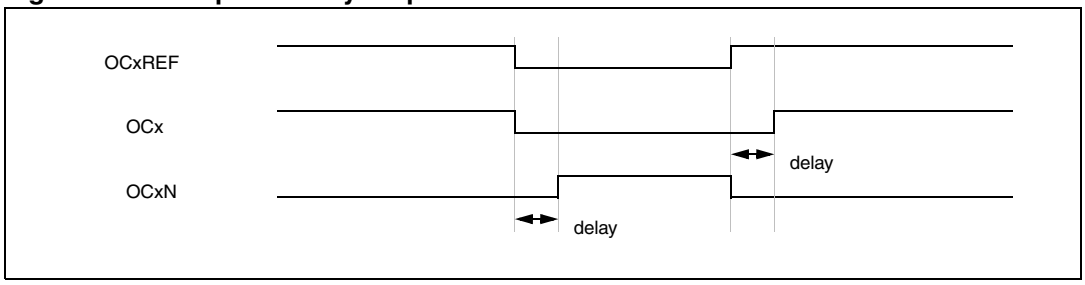


Figure 75. Dead-time waveforms with delay greater than the negative pulse.

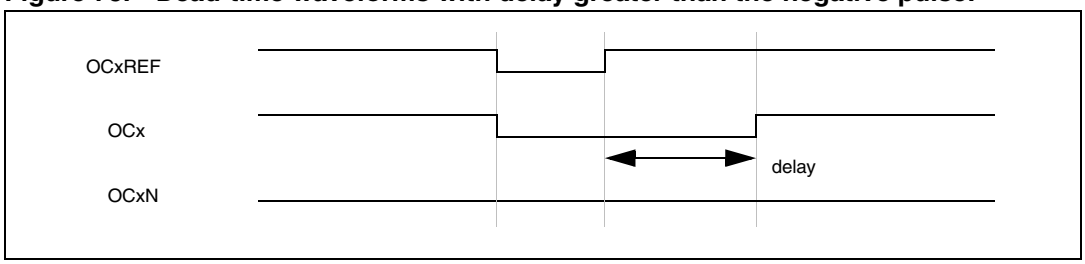
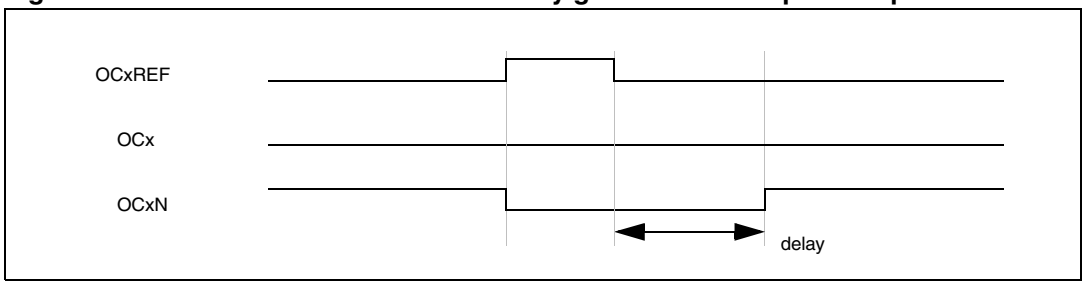


Figure 76. Dead-time waveforms with delay greater than the positive pulse.



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 12.4.18: TIM1 break and dead-time register (TIMx_BDTR) on page 268* 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 you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

12.3.12 Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to *Table 68: Output control bits for complementary OCx and OCxN channels with break feature on page 264* for more details.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller. For further information on the Clock Security System, refer to *Section 6.2.7: Clock security system (CSS)*.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function 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 you write MOE to 1 whereas it was low, you

must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read 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 in reset state (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 then the timer releases the enable output 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 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. A DMA request can be sent if the BDE 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 you write it to ‘1’ again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

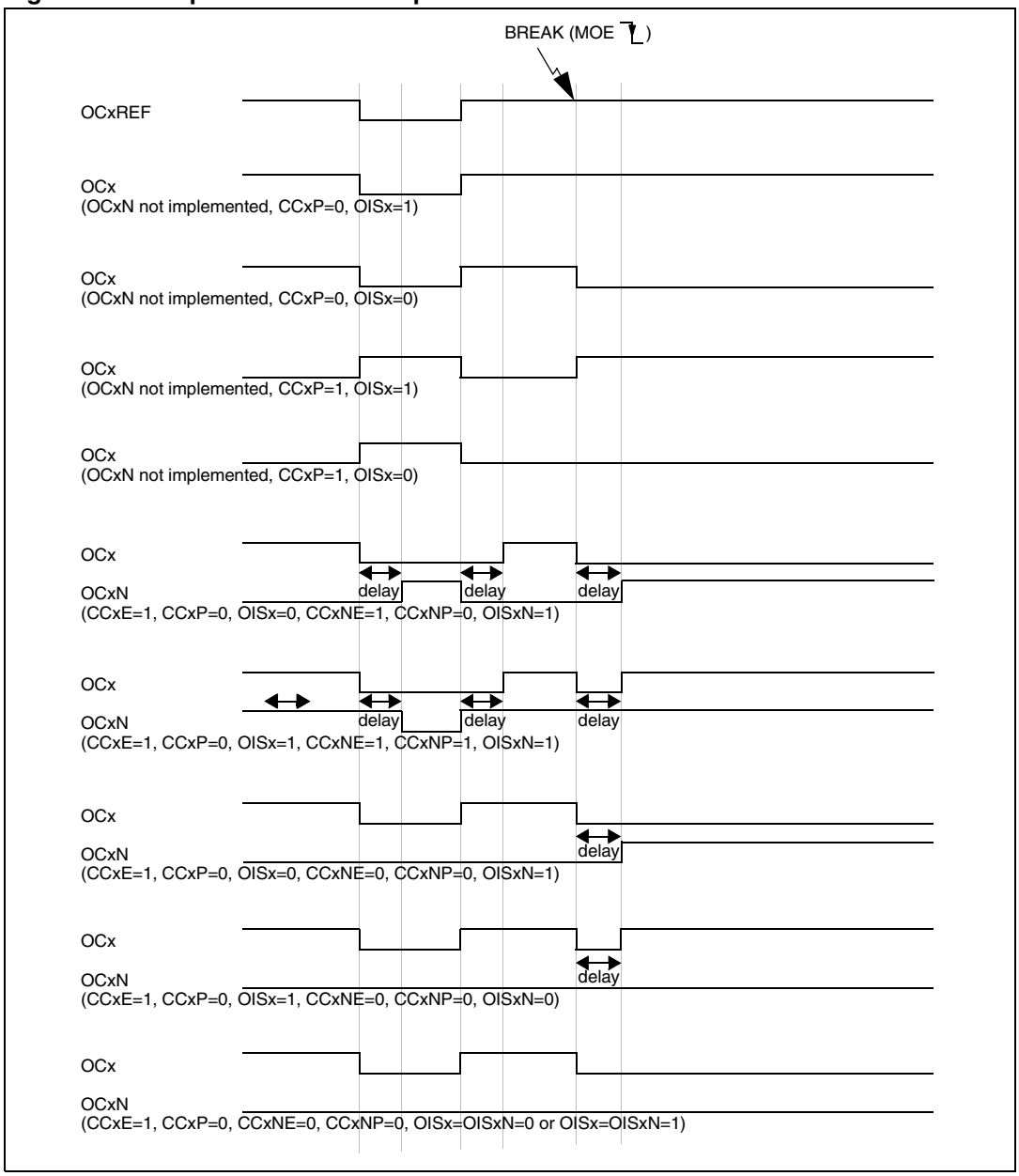
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 12.4.18: TIM1 break and dead-time register (TIMx_BDTR) on page 268*. The LOCK bits can be written only once after an MCU reset.

The *Figure 77* shows an example of behavior of the outputs in response to a break.

Figure 77. Output behavior in response to a break.



12.3.13 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to ‘1’). The OCxREF signal remains Low until the next update event, UEV, occurs.

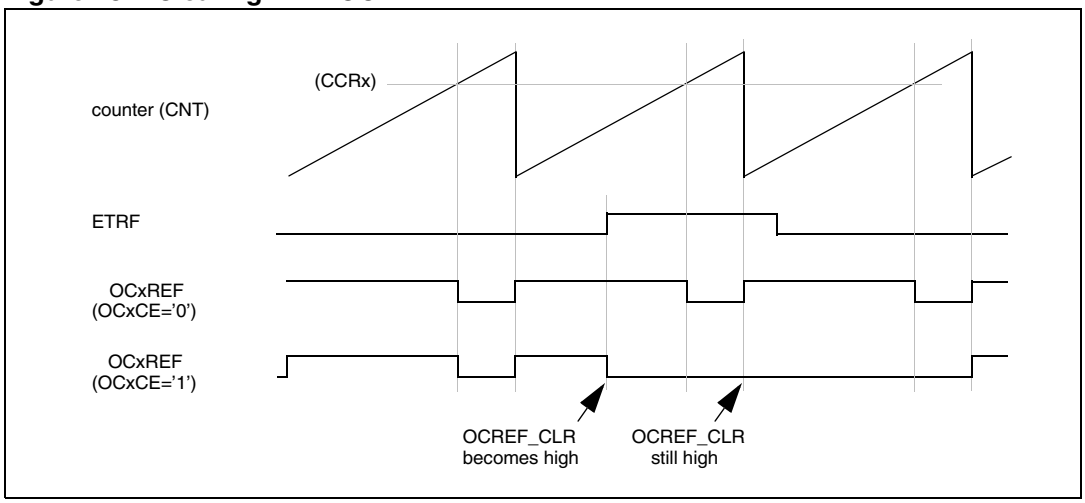
This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the OCxREF signal) can be connected to the output of a comparator to be used for current handling. In this case, the ETR must be configured as follow:

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 78 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 78. Clearing TIMx OCxREF



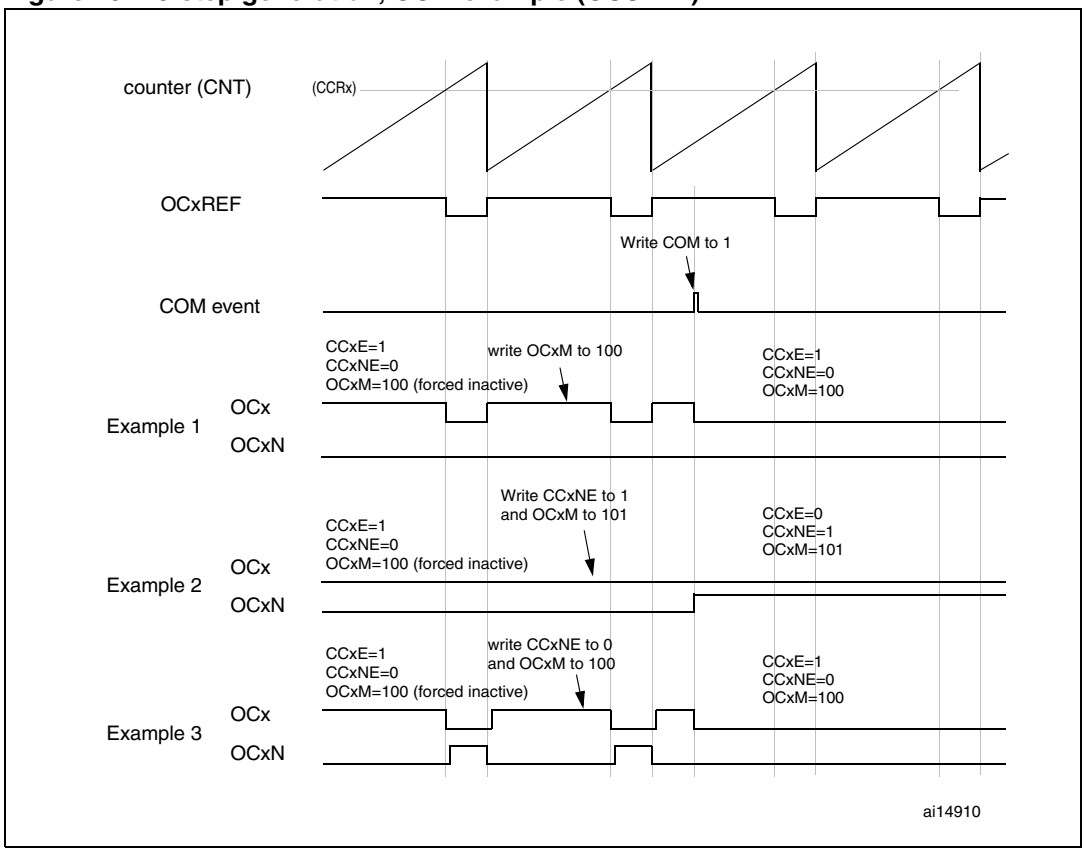
12.3.14 6-step PWM generation

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 you 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 79* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 79. 6-step generation, COM example (OSSR=1)



12.3.15 One-pulse mode

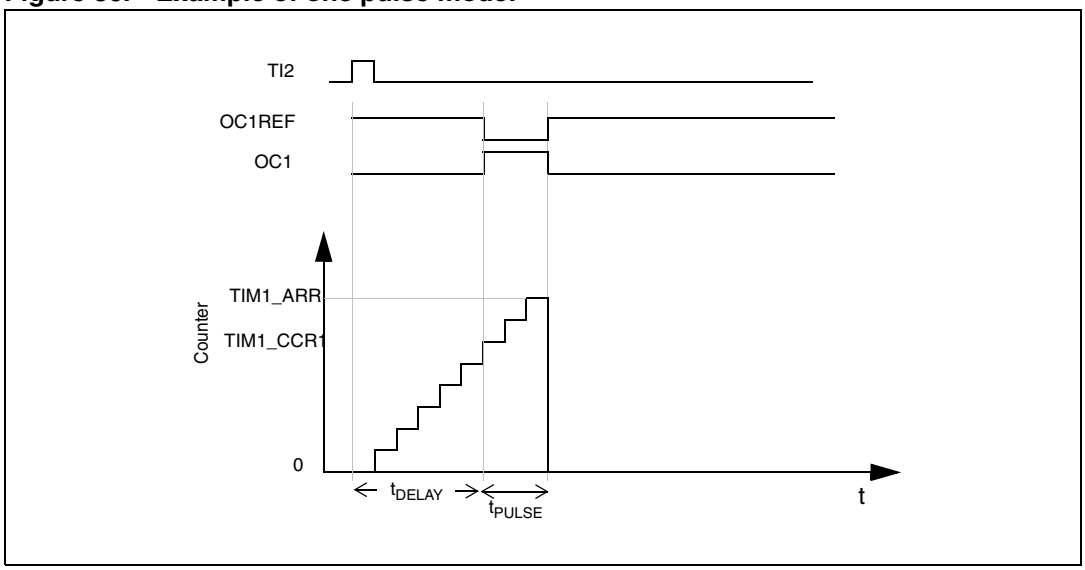
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode 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 ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 80. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
- 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_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you 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 you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have 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.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit 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.

12.3.16 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, you can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 66*. 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 you must configure TIMx_ARR before starting. in the same way, the capture, compare, prescaler,

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.

In this mode, the counter is modified automatically following the speed and the direction of the incremental 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 don't switch at the same time.

Table 66. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 81* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='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).

Figure 81. Example of counter operation in encoder interface mode.

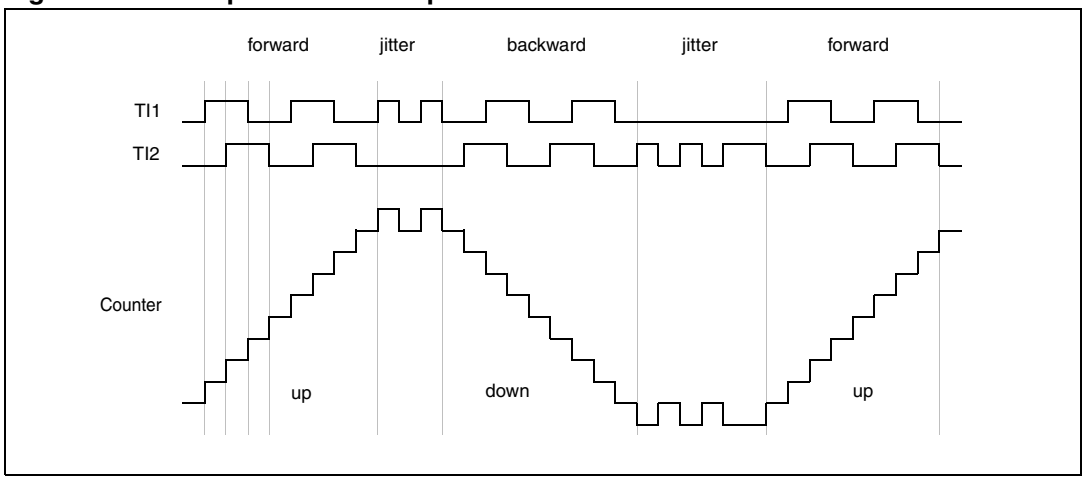
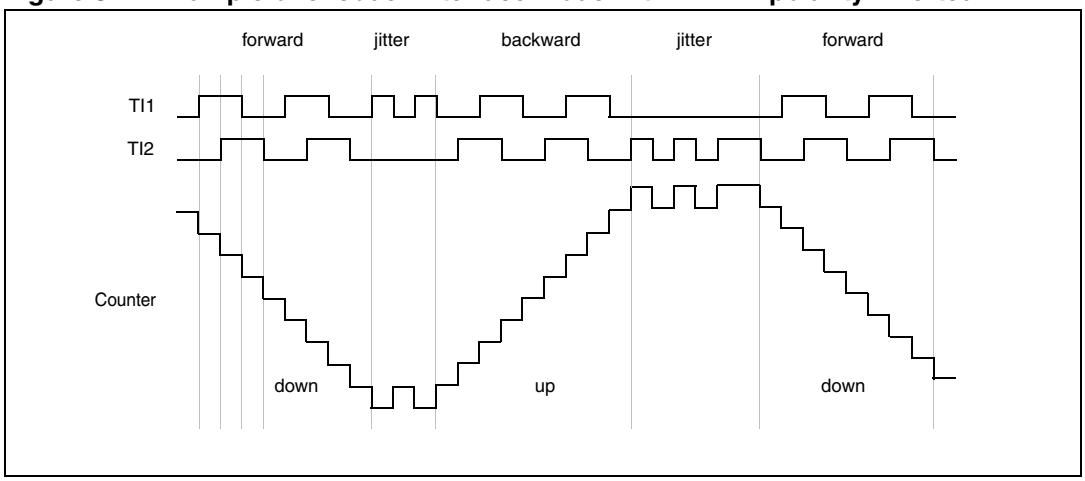


Figure 82 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 82. 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. You can obtain dynamic information (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. You can do this 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.

12.3.17 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 a 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. An example of this feature used to interface Hall sensors is given in *Section 12.3.18* below.

12.3.18 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4 or TIM5) referred to as “interfacing timer” in *Figure 83*. 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 66: Capture/compare channel (example: channel 1 input stage) on page 223*). 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: you want to change the PWM configuration of your advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

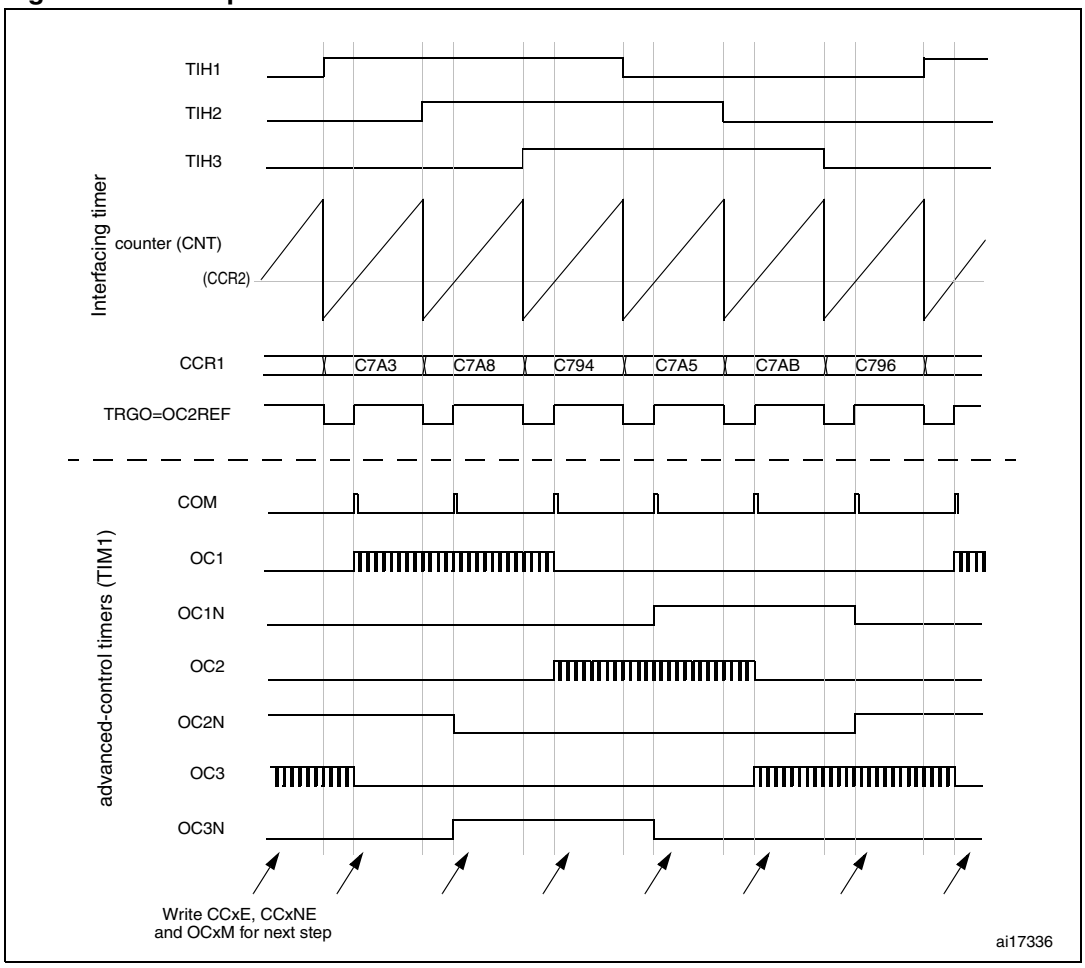
- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 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’. You can also program the digital filter 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 83* describes this example.

Figure 83. Example of hall sensor interface



12.3.19 TIMx and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

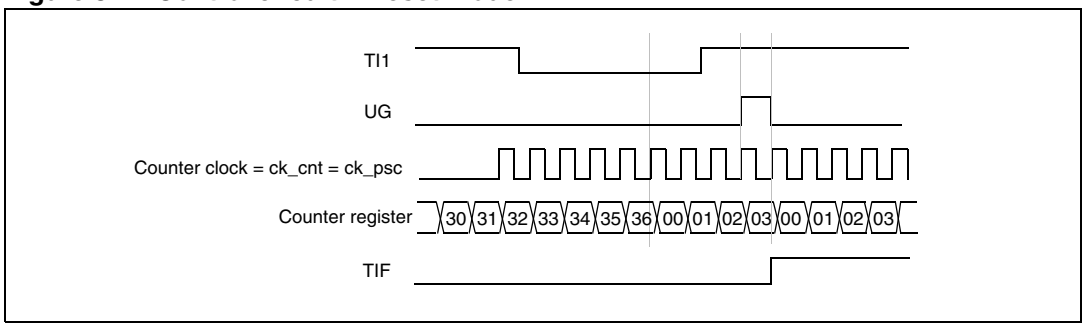
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=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=101 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 84. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

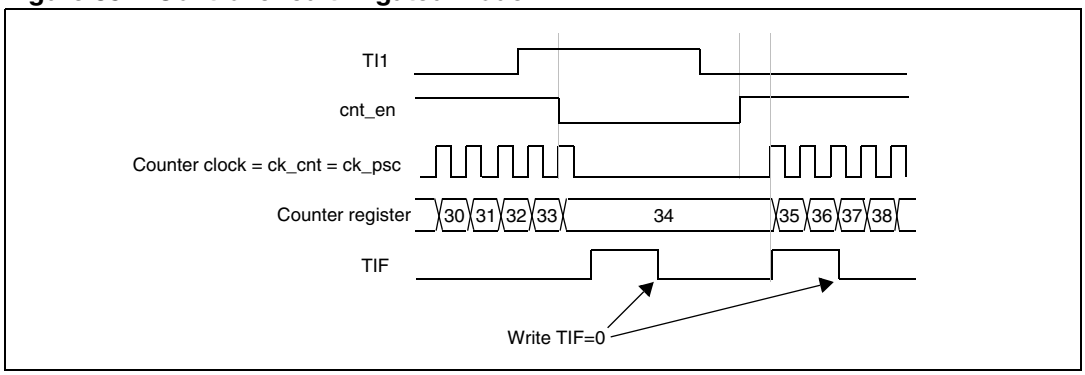
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 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=101 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 85. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

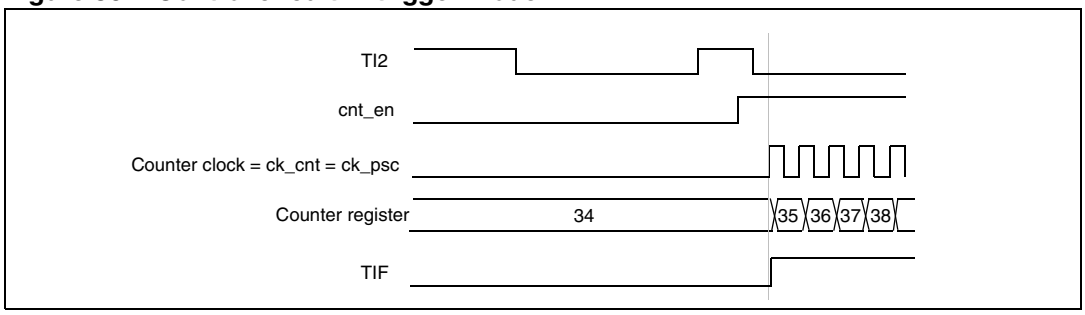
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 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=110 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 86. Control circuit in trigger mode



Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (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:

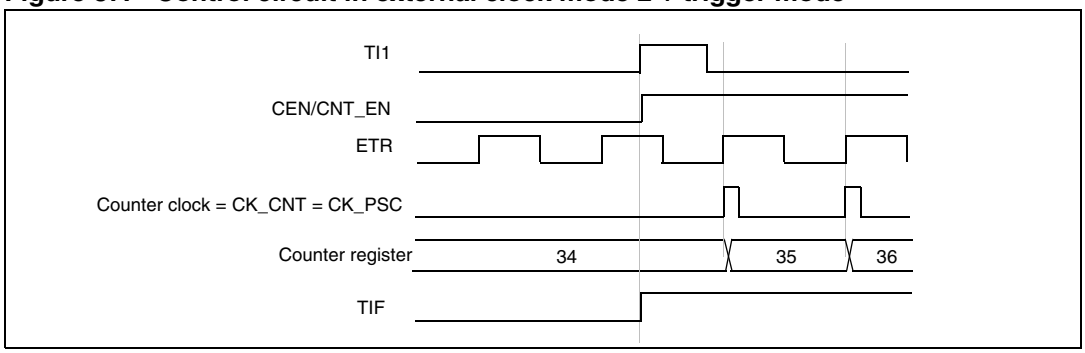
1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.

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=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=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=101 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 87. Control circuit in external clock mode 2 + trigger mode



12.3.20 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 13.3.15: Timer synchronization on page 305* for details.

12.3.21 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 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 25.15.2: Debug support for timers, watchdog and I2C*.

12.4 TIM1 registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

12.4.1 TIM1 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, always read as 0

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 (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
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 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

12.4.2 TIM1 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 15 Reserved, always read as 0

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)
refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)
refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)
refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[1:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, always read as 0

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when

Note: This bit acts only on channels that have a complementary output.

12.4.3 TIM1 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: **1:** Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).*

***2:** It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).*

***3:** If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N 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

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 6:4 **TS[2:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0)
001: Internal Trigger 1 (ITR1)
010: Internal Trigger 2 (ITR2)
011: Internal Trigger 3 (ITR3)
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)
111: External Trigger input (ETRF)

See *Table 67: TIMx Internal trigger connection on page 253* 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 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** 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.
000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.
001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
101: 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.
110: 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.
111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 67. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM5 or TIM15(1)	TIM2	TIM3	TIM4

1. TIM5 only in high density value line devices. Selection of TIM5 or TIM15 depends on the MISC_REMAP bit in the AFIO_MAPR2 register.

12.4.4 TIM1 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	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

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

12.4.5 TIM1 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred.
1: An active level has been detected on the break input.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode.It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.
0: No COM event occurred.
1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 12.4.3: TIM1 slave mode control register (TIMx_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

12.4.6 TIM1 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware
0: No action
1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation

refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

12.4.7 TIM1 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). 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. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit 15 **OC2CE:** Output Compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: 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

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 6:4 **OC1M**: 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.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: 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).
010: 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).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: 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’).
111: 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.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).*

***2:** In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently 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**: 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).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: 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 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**: 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**: 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).

12.4.8 TIM1 capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC4CE:** Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: 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 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: 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).

Input capture mode

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: 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**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: 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).

12.4.9 TIM1 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:14 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
0: OC1N active high.
1: OC1N active low.
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).

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
This bit selects whether IC1 or IC1 is used for trigger or capture operations.
0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted.
1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted.
Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **CC1E**: Capture/Compare 1 output enable
CC1 channel configured as output:
0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Table 68. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
		1	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), 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 and AFIO registers.

12.4.10 TIM1 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

12.4.11 TIM1 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

12.4.12 TIM1 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 12.3.1: Time-base unit on page 210* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

12.4.13 TIM1 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

12.4.14 TIM1 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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).

12.4.15 TIM1 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 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 signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

12.4.16 TIM1 capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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_CCMR3 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).

12.4.17 TIM1 capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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_CCMR4 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).

12.4.18 TIM1 break and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

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.

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 12.4.9: TIM1 capture/compare enable register (TIMx_CCER) on page 262*).

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).

Note: 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).

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 12.4.9: TIM1 capture/compare enable register (TIMx_CCER) on page 262*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0).
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1. Then, 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).

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 12.4.9: TIM1 capture/compare enable register (TIMx_CCER) on page 262*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0).
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected.
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

12.4.19 TIM1 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, always read as 0

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 t
the TIMx_DMAR address)
00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
10001: 18 transfers

Bits 7:5 Reserved, always read as 0

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

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.

12.4.20 TIM1 DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
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).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) 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

Note: 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.

12.4.21 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 69. TIM1 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIMx_DIER	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0		0	0	0	0	0	0	0	0
0x14	TIMx_EGR	Reserved																								BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0	0	0	0	0	0	0	0
0x18	TIMx_CCMR1 Output Compare mode	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	TIMx_CCMR2 Output Compare mode	Reserved																O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Reserved																IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Reserved																		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TIMx_RCR	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 69. TIM1 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
0x34	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Reserved																CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Reserved																CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Reserved																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Reserved																			DBL[4:0]					Reserved		DBA[4:0]				
	Reset value																				0	0	0	0	0			0	0	0	0	0
0x4C	TIMx_DMAR	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

13 General-purpose timers (TIM2 to TIM5)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

13.1 TIM2 to TIM5 introduction

The general-purpose 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 and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

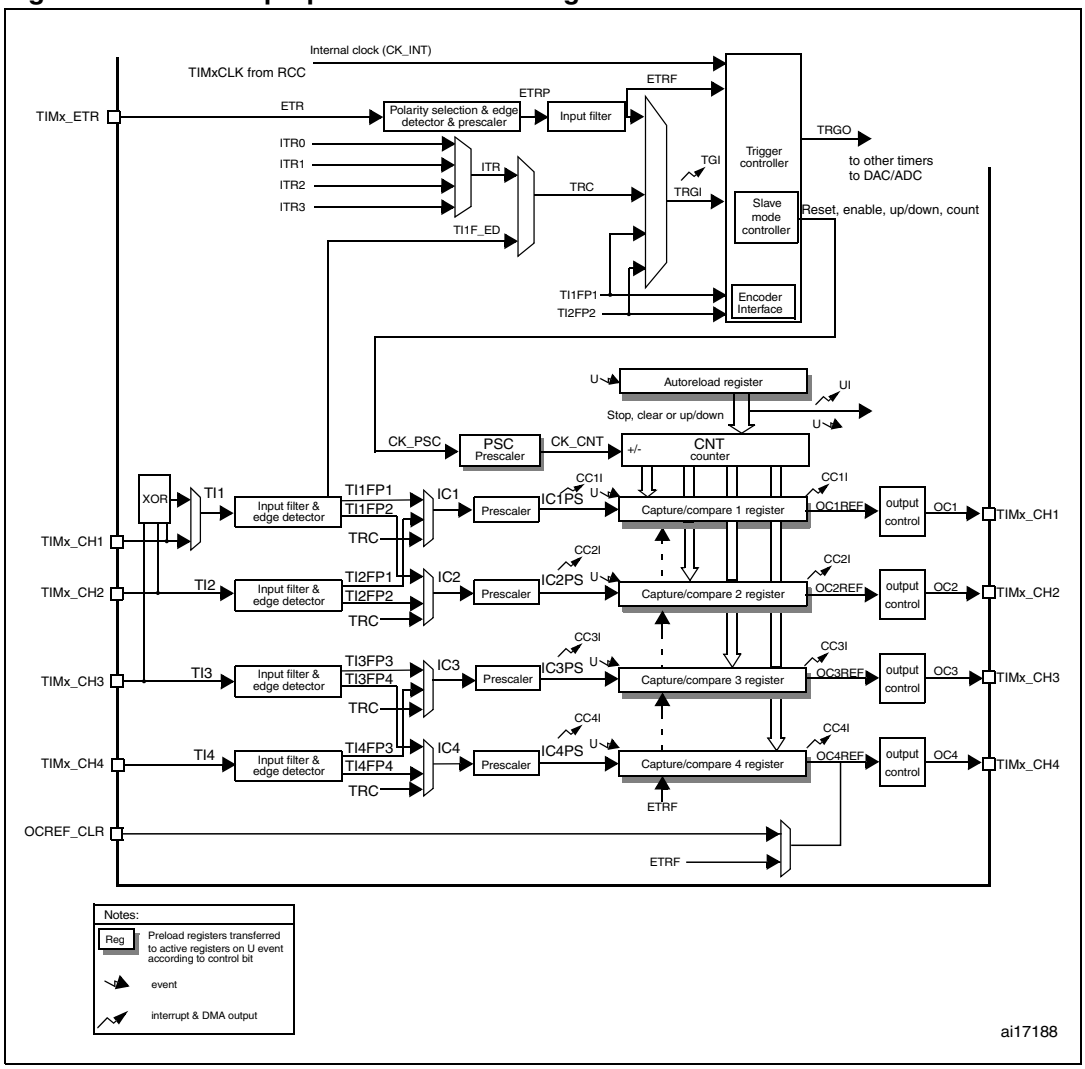
The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 13.3.15*.

13.2 TIM2 to TIM5 main features

General-purpose TIMx timer features include:

- 16-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 88. General-purpose timer block diagram



13.3 TIM2 to TIM5 functional description

13.3.1 Time-base unit

The main block of the programmable timer is a 16-bit counter with its related auto-reload register. The counter can count up but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit 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 89 and *Figure 90* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 89. Counter timing diagram with prescaler division change from 1 to 2

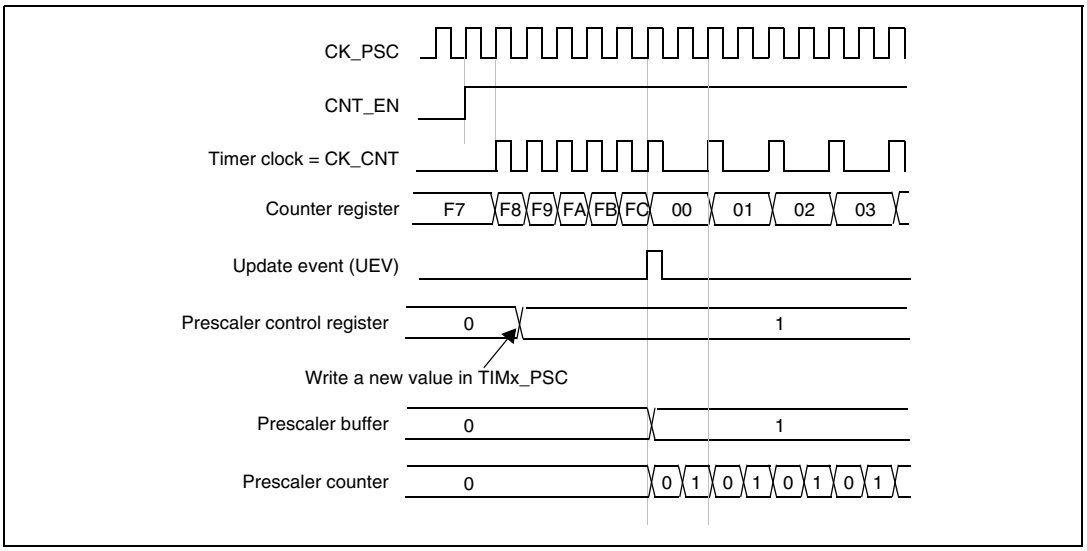
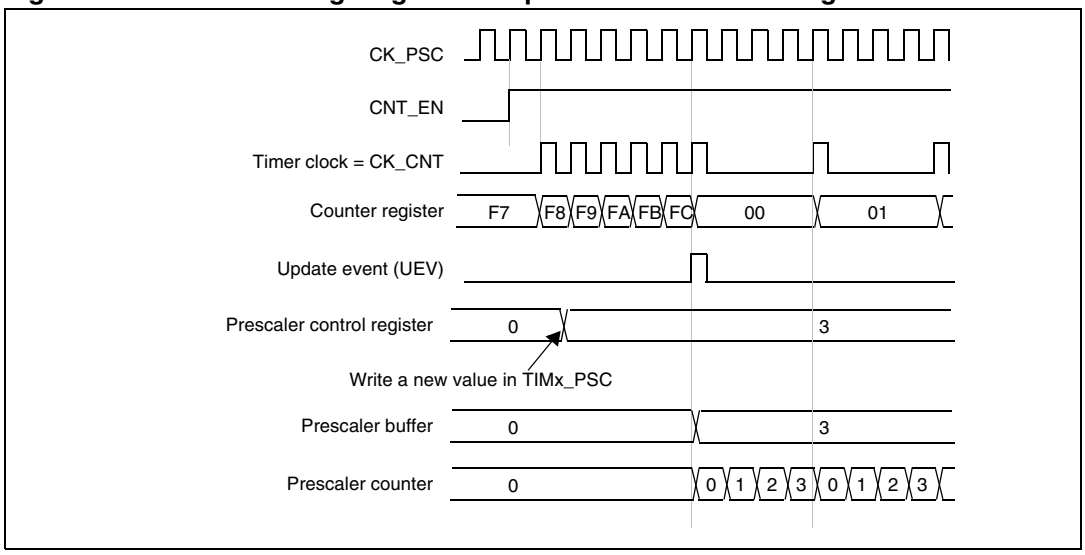


Figure 90. Counter timing diagram with prescaler division change from 1 to 4



13.3.2 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 91. Counter timing diagram, internal clock divided by 1

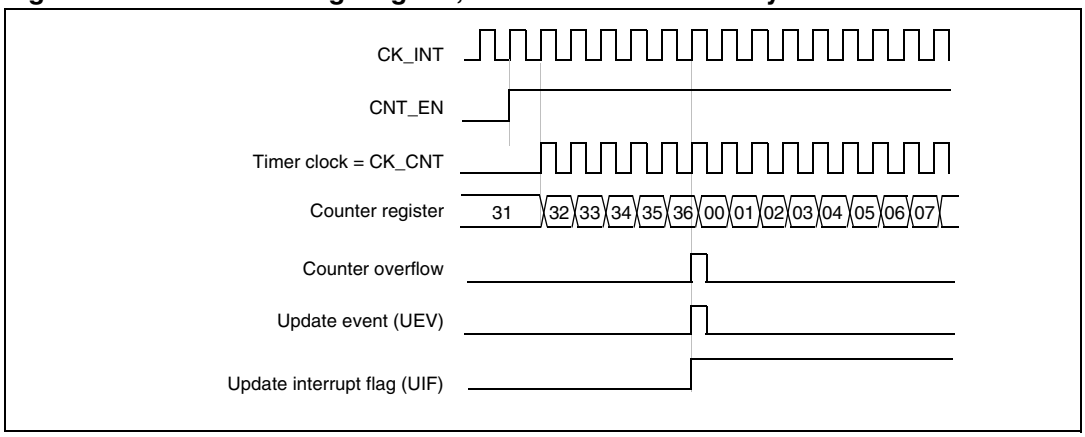


Figure 92. Counter timing diagram, internal clock divided by 2

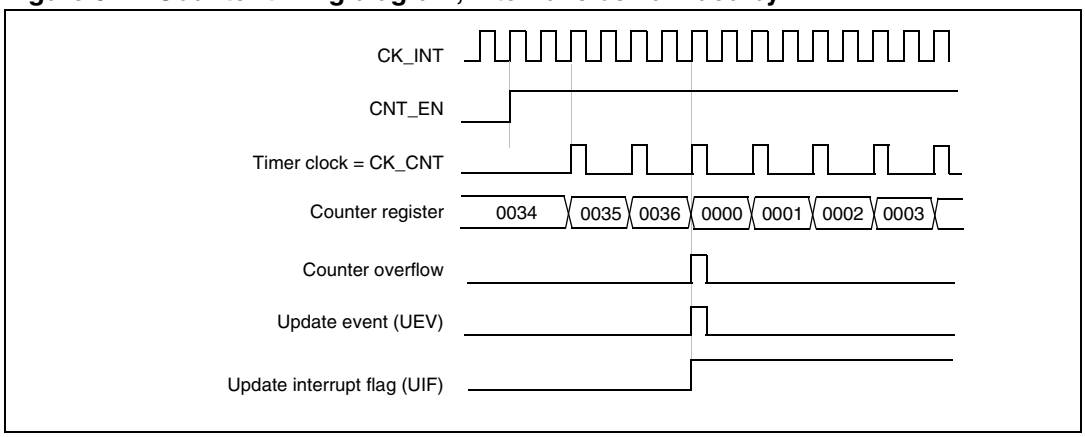


Figure 93. Counter timing diagram, internal clock divided by 4

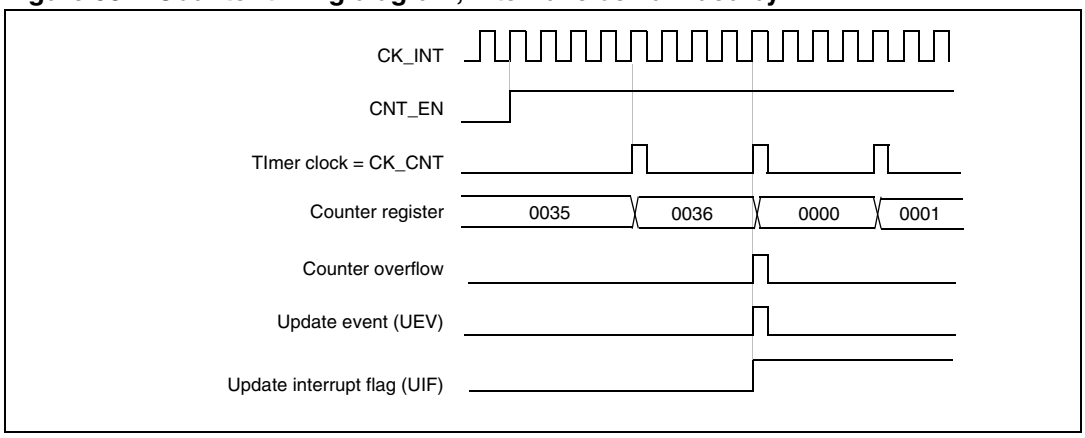


Figure 94. Counter timing diagram, internal clock divided by N

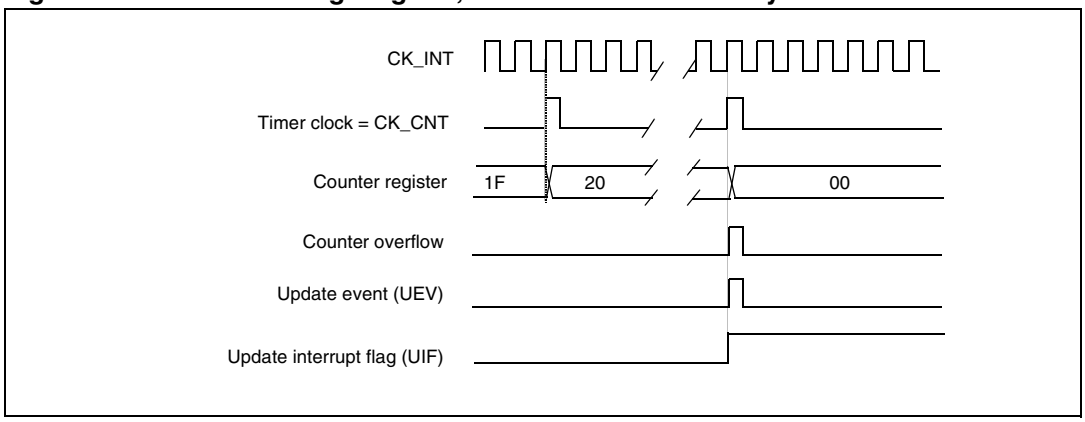


Figure 95. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

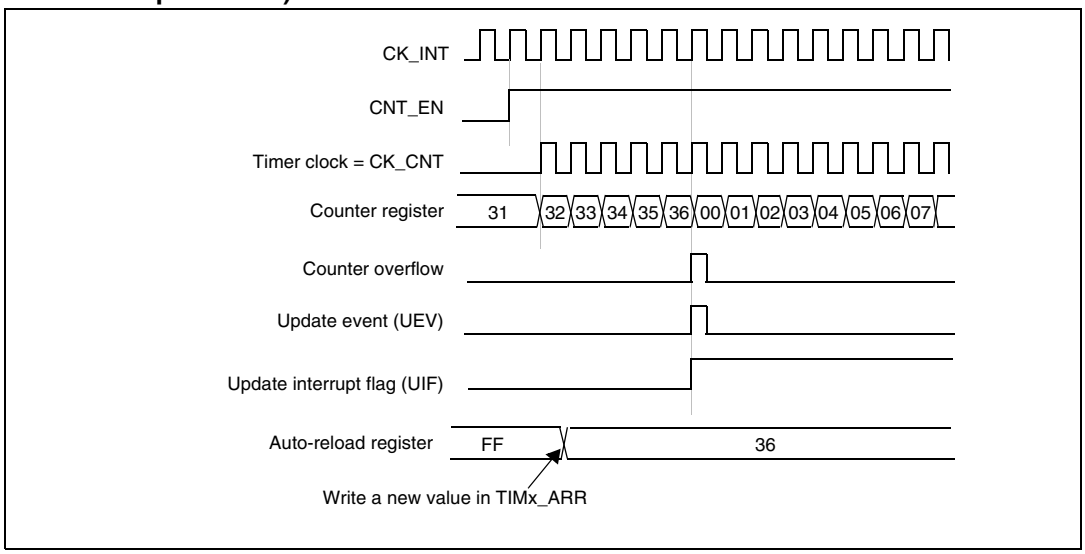
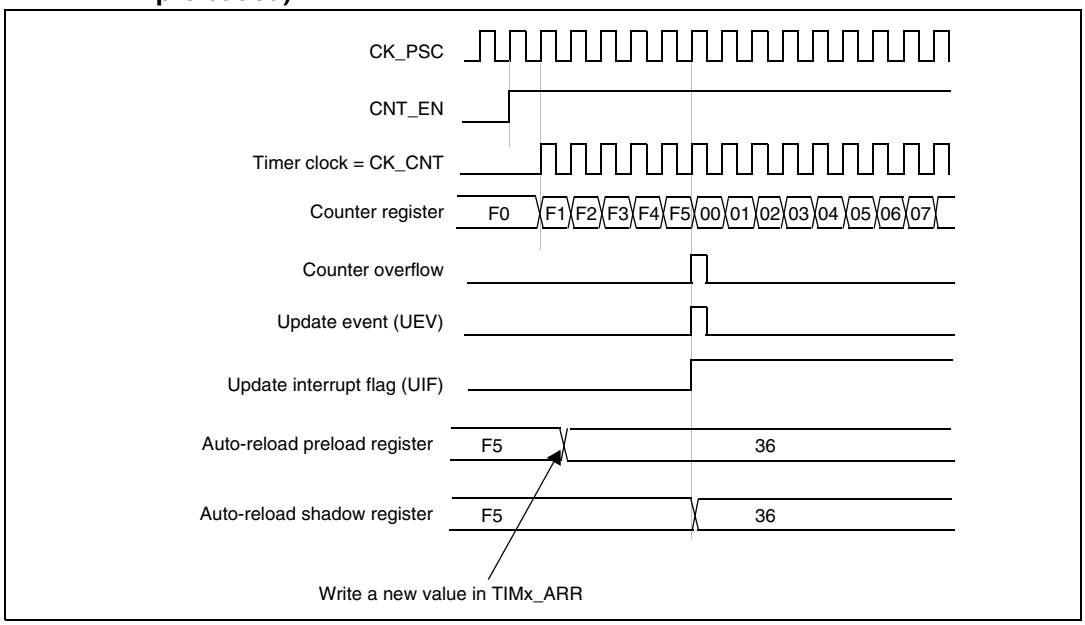


Figure 96. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



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.

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 97. Counter timing diagram, internal clock divided by 1

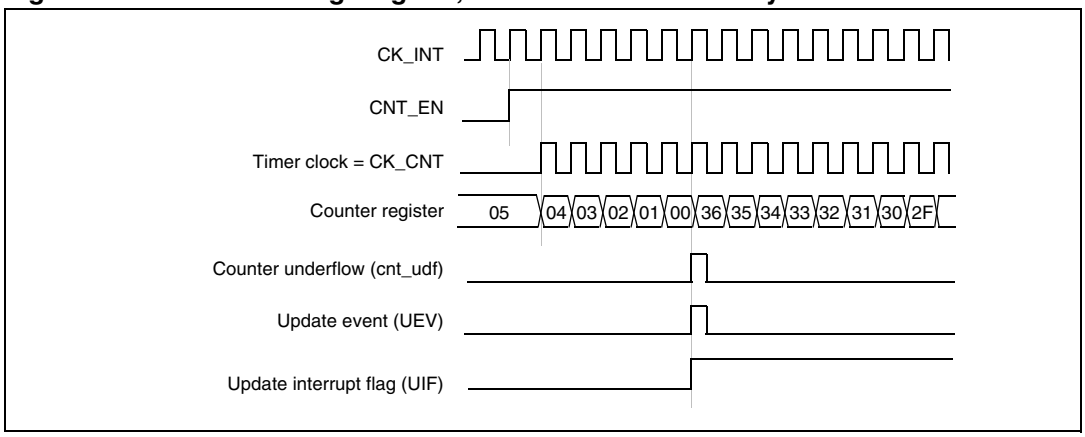


Figure 98. Counter timing diagram, internal clock divided by 2

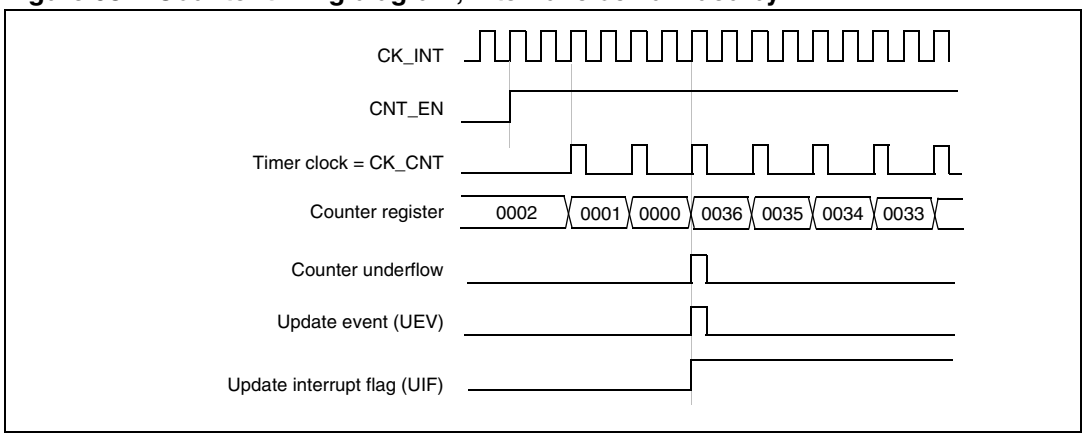


Figure 99. Counter timing diagram, internal clock divided by 4

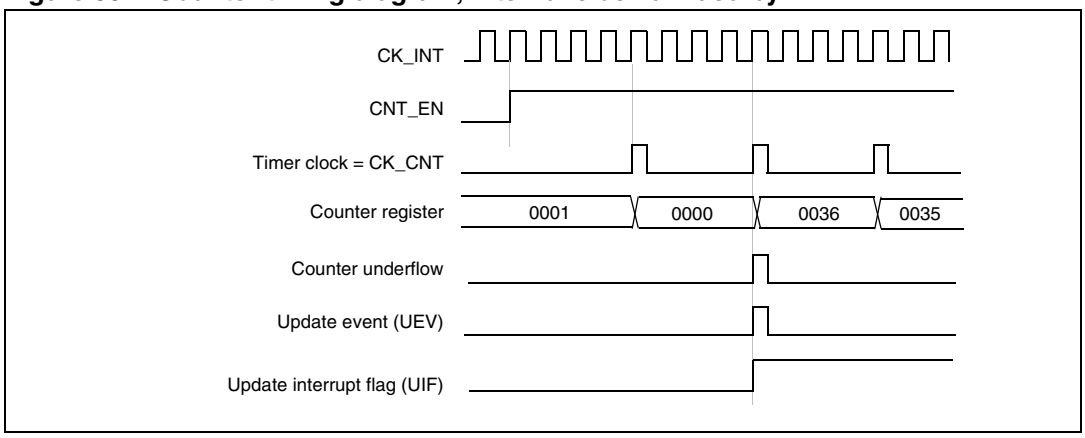


Figure 100. Counter timing diagram, internal clock divided by N

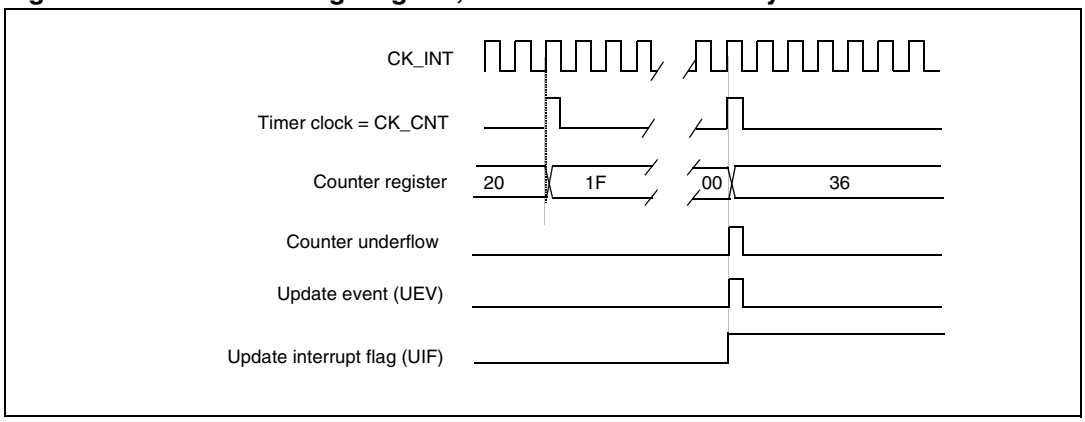
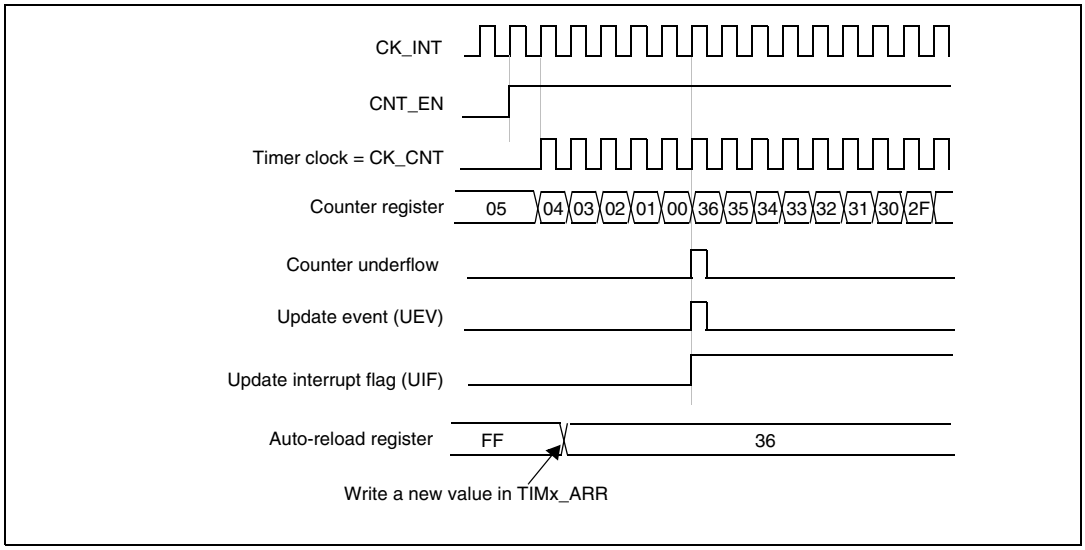


Figure 101. Counter timing diagram, Update event when repetition counter is not used



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.

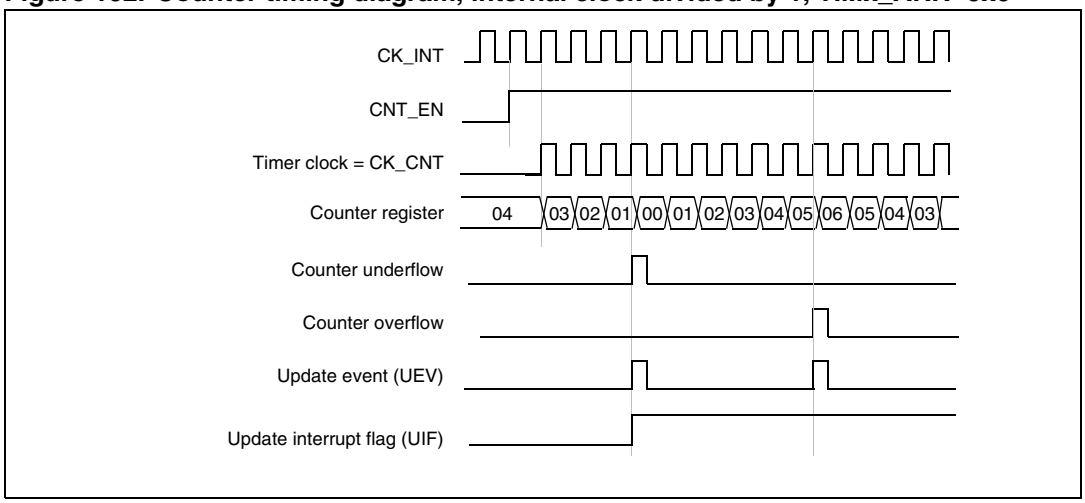
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on 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 102. 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 13.4.1: TIMx control register 1 (TIMx_CR1) on page 311*).

Figure 103. Counter timing diagram, internal clock divided by 2

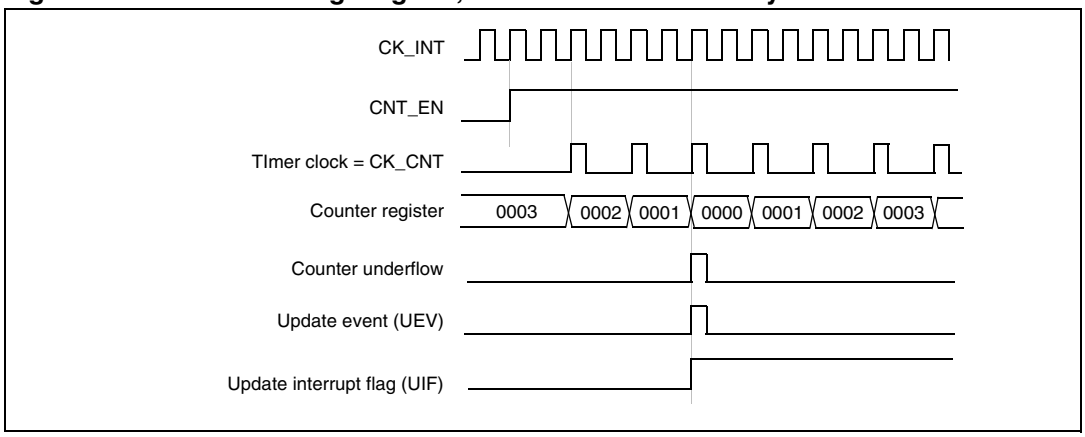
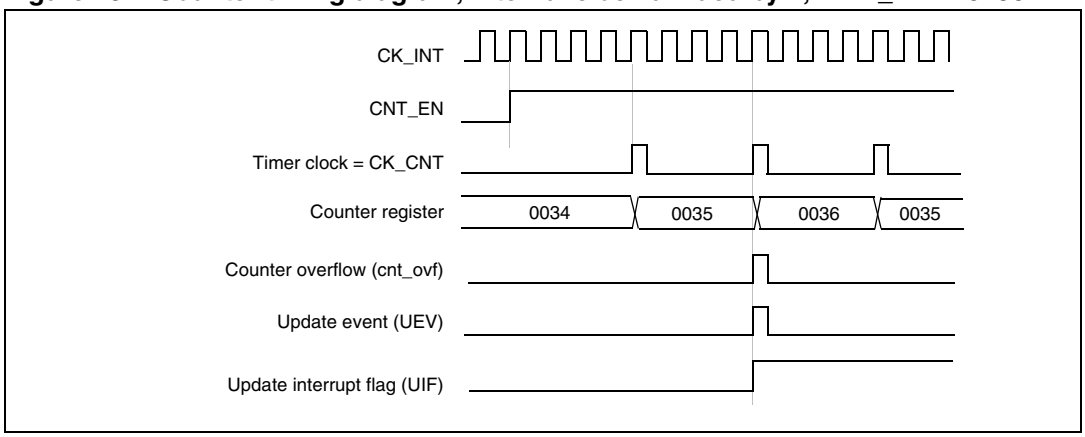


Figure 104. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 105. Counter timing diagram, internal clock divided by N

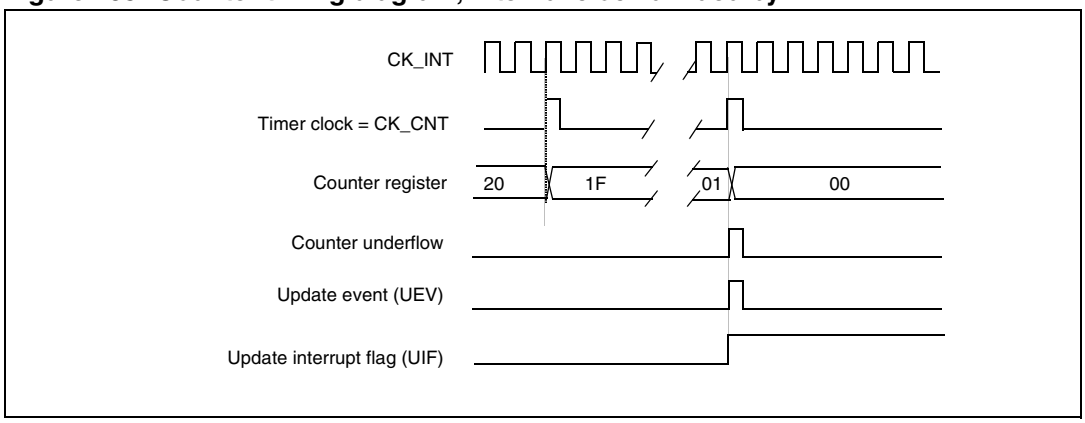


Figure 106. Counter timing diagram, Update event with ARPE=1 (counter underflow)

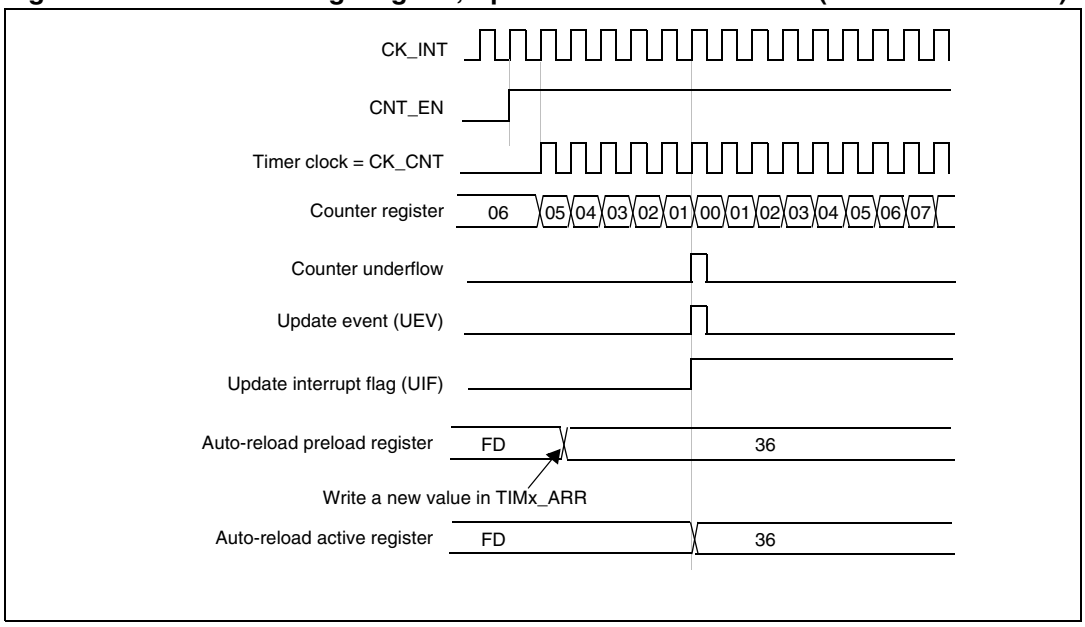
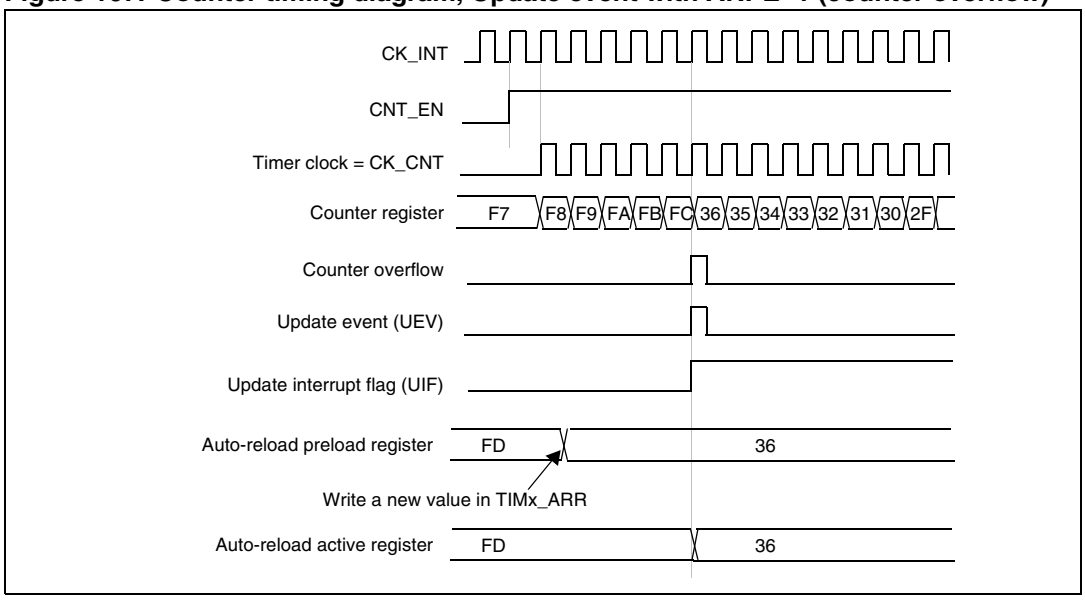


Figure 107. Counter timing diagram, Update event with ARPE=1 (counter overflow)



13.3.3 Clock selection

The counter clock can be provided by the following clock sources:

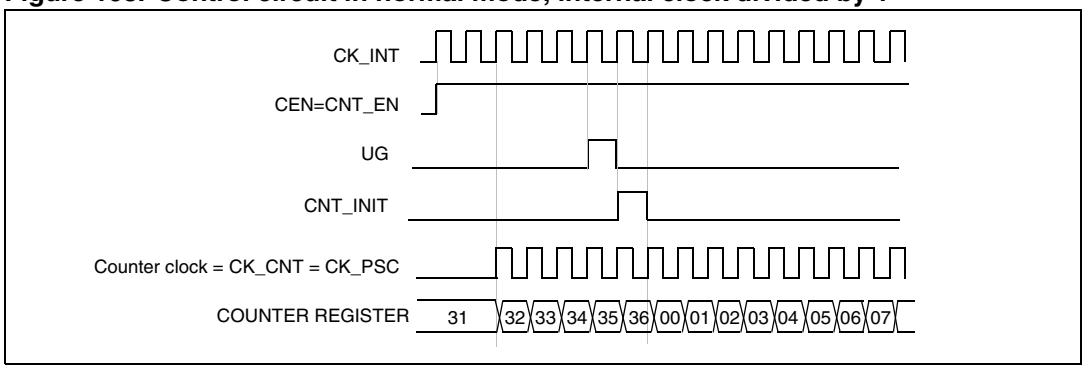
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer 1 to act as a prescaler for Timer 2. Refer to *: Using one timer as prescaler for another on page 306* 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 108 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

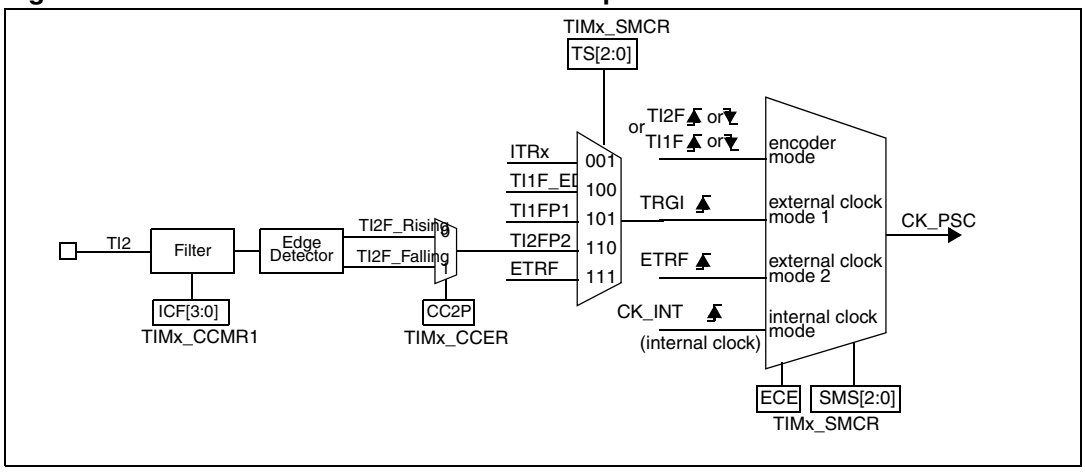
Figure 108. Control circuit in normal mode, internal clock divided by 1



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.

Figure 109. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
2. 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).

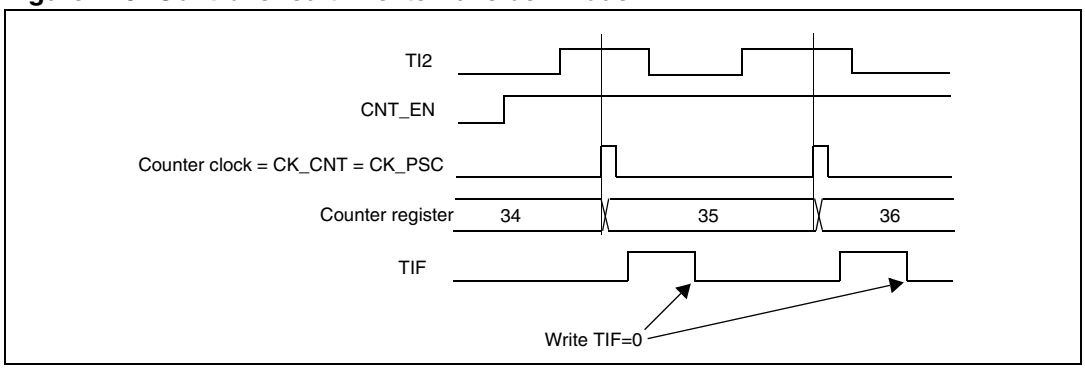
Note: The capture prescaler is not used for triggering, so you don’t need to configure it.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 110. Control circuit in external clock mode 1



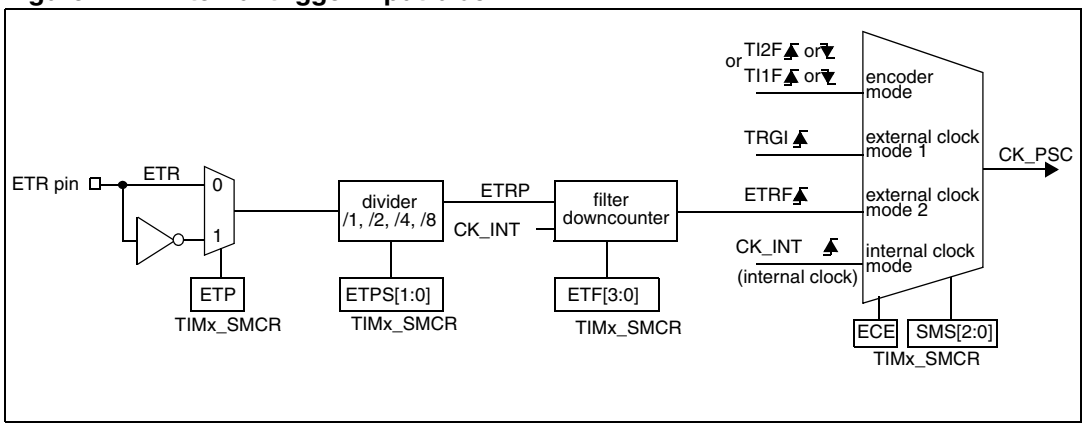
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 111* gives an overview of the external trigger input block.

Figure 111. External trigger input block



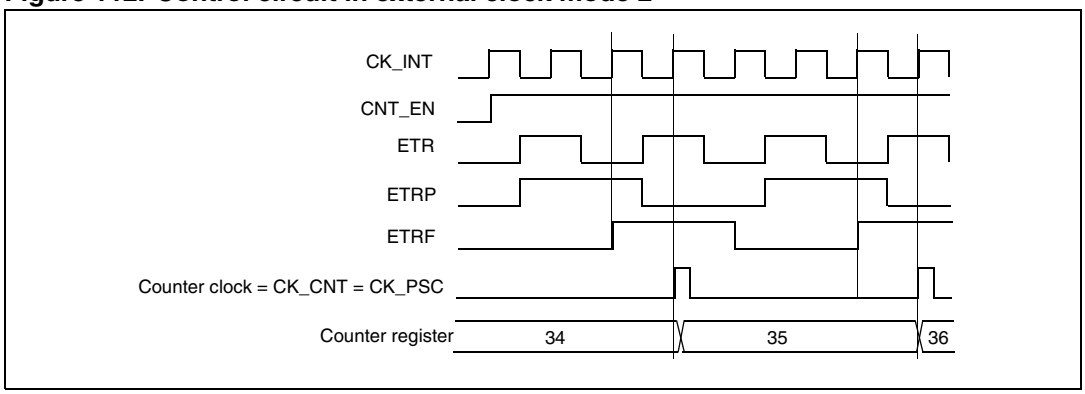
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 112. Control circuit in external clock mode 2



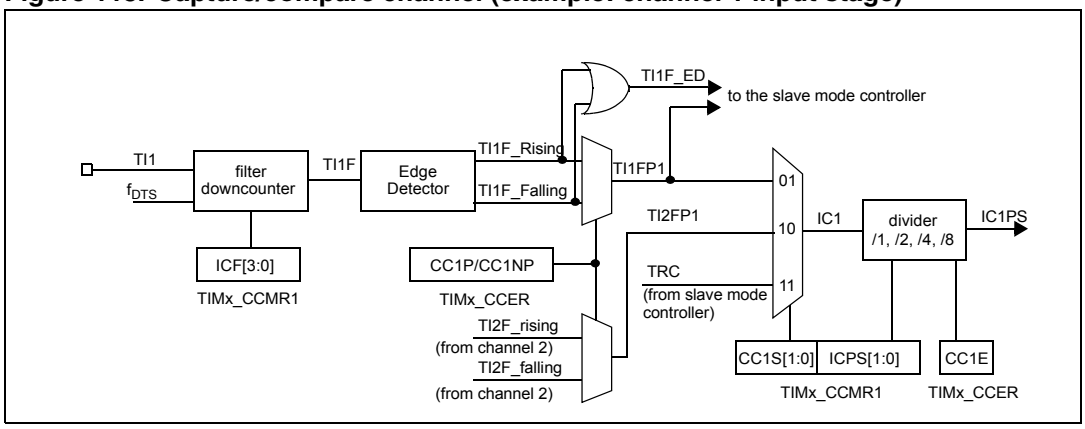
13.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 113. 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 114. Capture/compare channel 1 main circuit

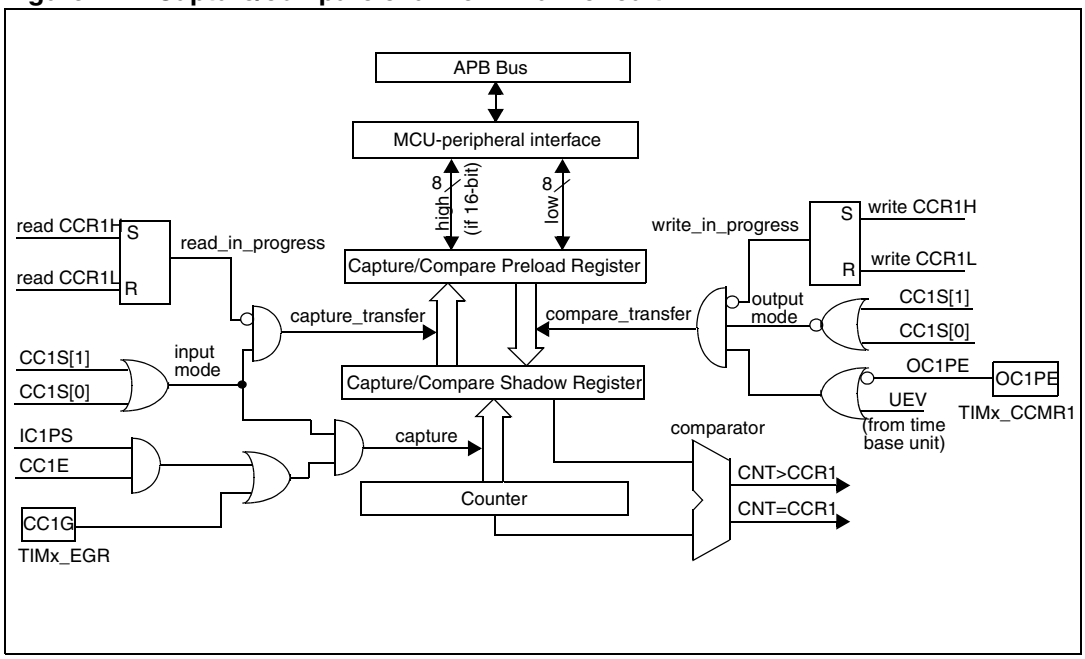
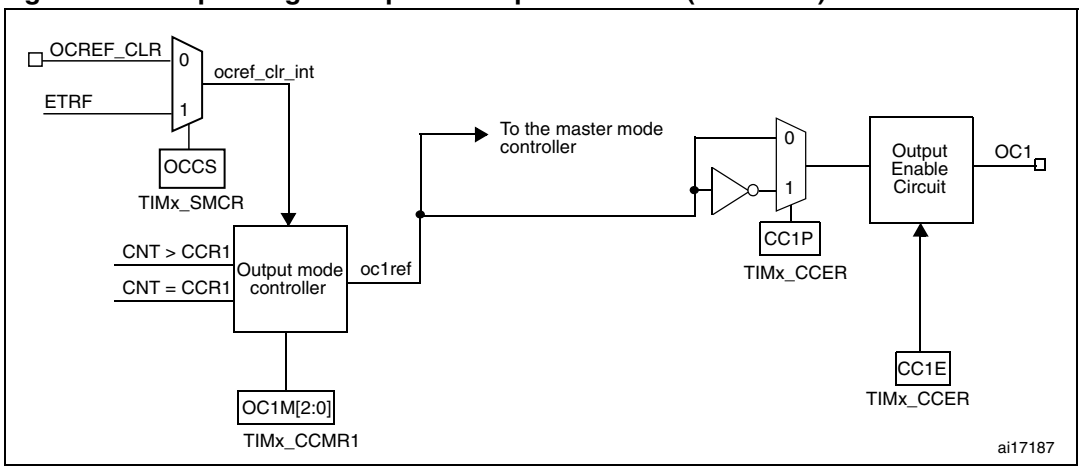


Figure 115. Output stage of capture/compare channel (channel 1)



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.

13.3.5 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 you write it to 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:

- 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.
- Program the input filter duration you need with respect to the signal you connect 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 f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the TI1 channel by writing the CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- 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).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- 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.

13.3.6 PWM input mode

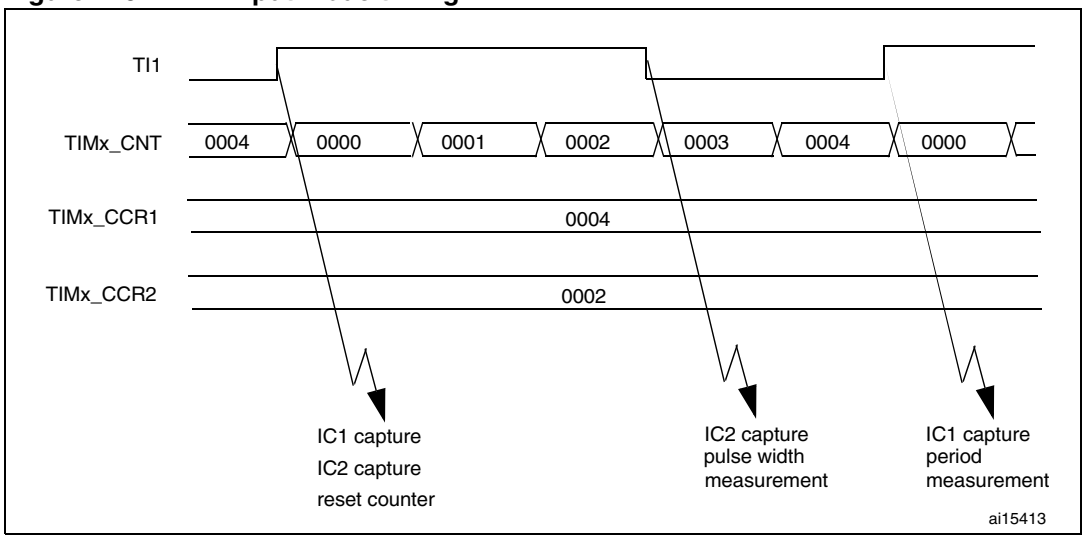
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you 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):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1 in the TIMx_CCER register.

Figure 116. PWM input mode timing



13.3.7 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, you just need 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.

13.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

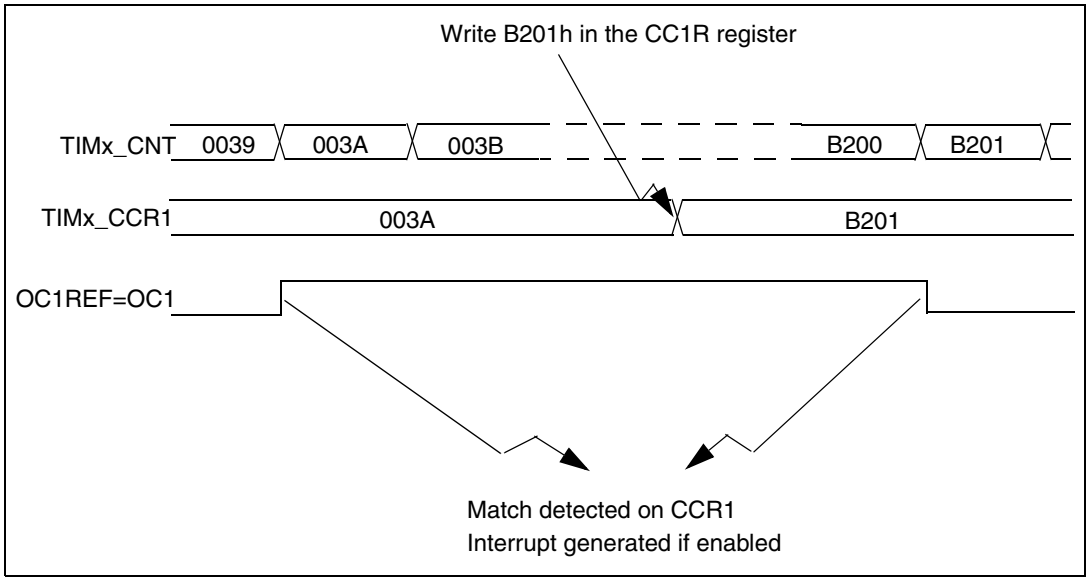
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, you must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 117*.

Figure 117. Output compare mode, toggle on OC1.



13.3.9 PWM mode

Pulse width modulation mode allows you 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. You must enable the corresponding preload register 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, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx≤ TIMx_CNT or TIMx_CNT≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison changes, 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.

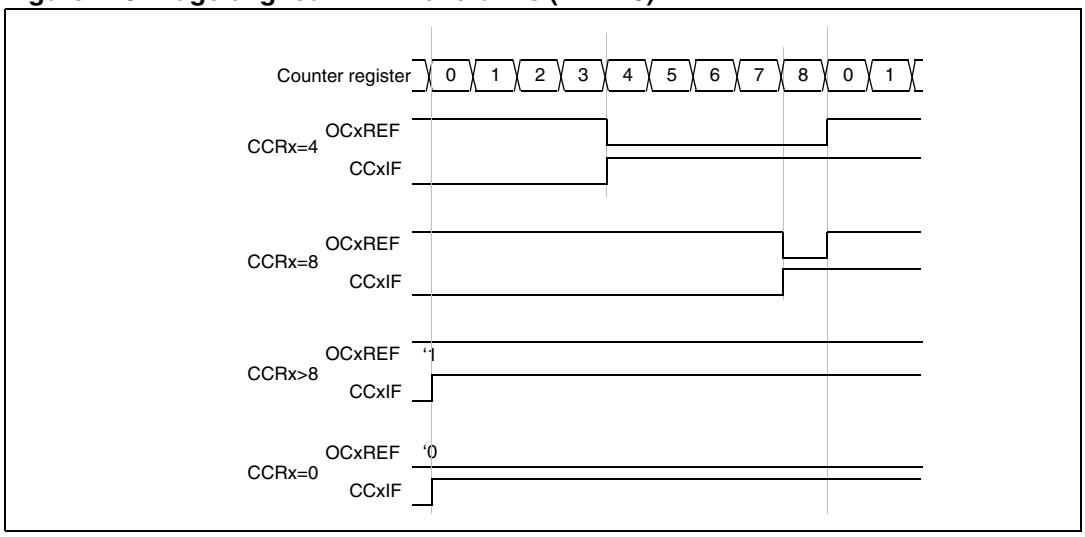
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Section : Upcounting mode on page 278*.

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 118* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 118. 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 281*

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.

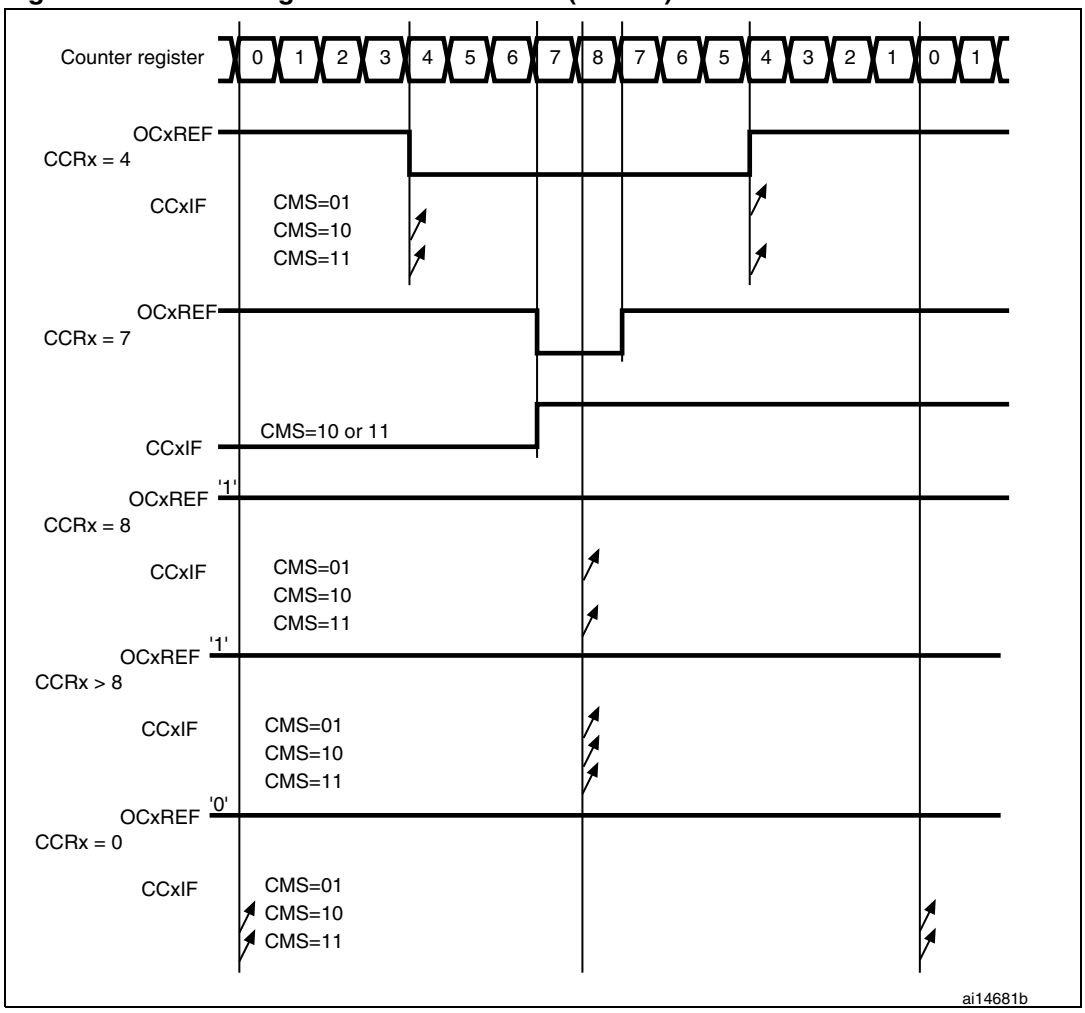
PWM center-aligned mode

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 283*.

Figure 119 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 119. Center-aligned PWM waveforms (ARR=8)



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 you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value 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.

13.3.10 One-pulse mode

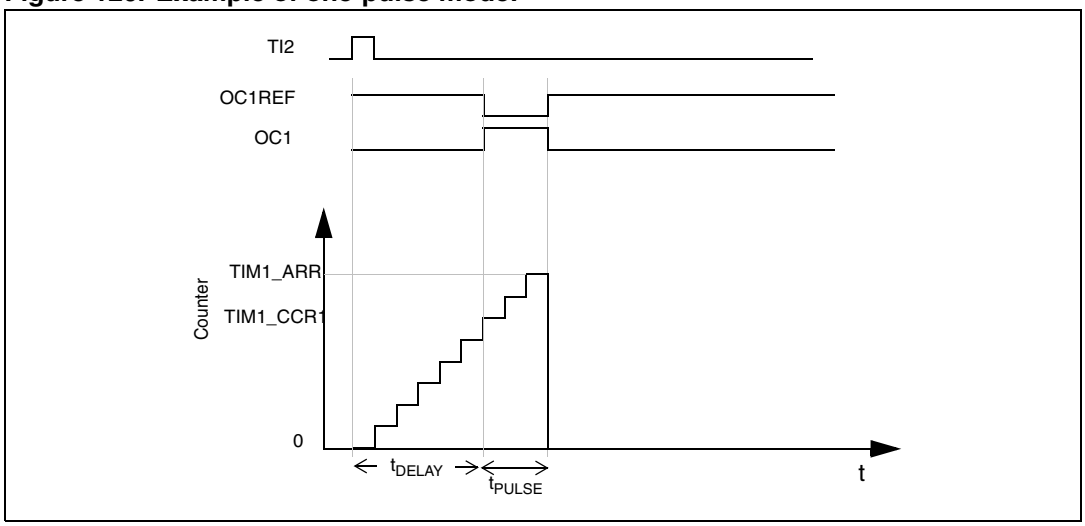
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode 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≤ ARR (in particular, 0<CCRx),
- In downcounting: CNT>CCRx.

Figure 120. Example of one-pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 on TI2 by writing IC2S=01 in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P=0 in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
- 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_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you 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 you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have 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.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit 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.

13.3.11 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next update event (UEV) occurs.

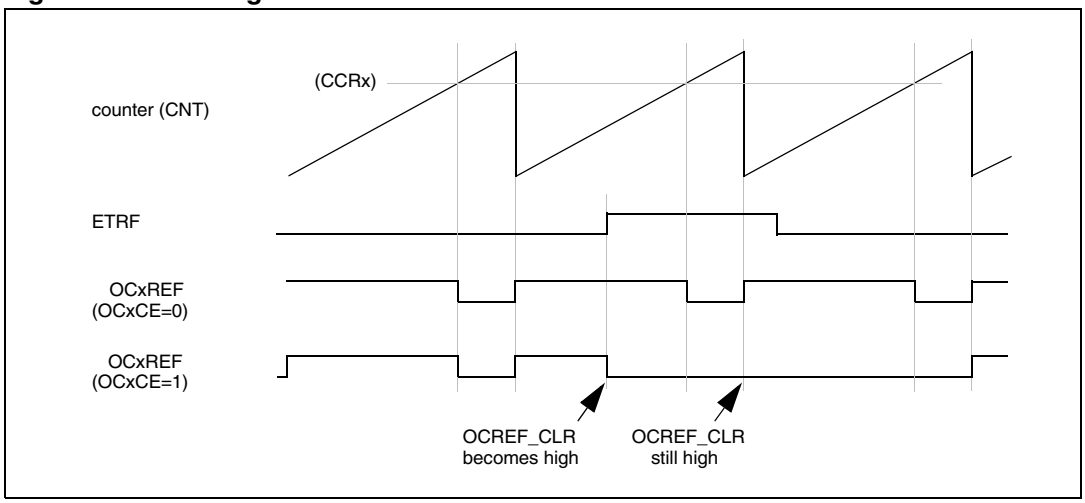
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application’s needs.

Figure 121 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 121. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

13.3.12 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, you can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 70*. 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 you must configure TIMx_ARR 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 incremental 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 don't switch at the same time.

Table 70. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 122 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P=0, CC1NP = '0 (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P=0, 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 122. Example of counter operation in encoder interface mode

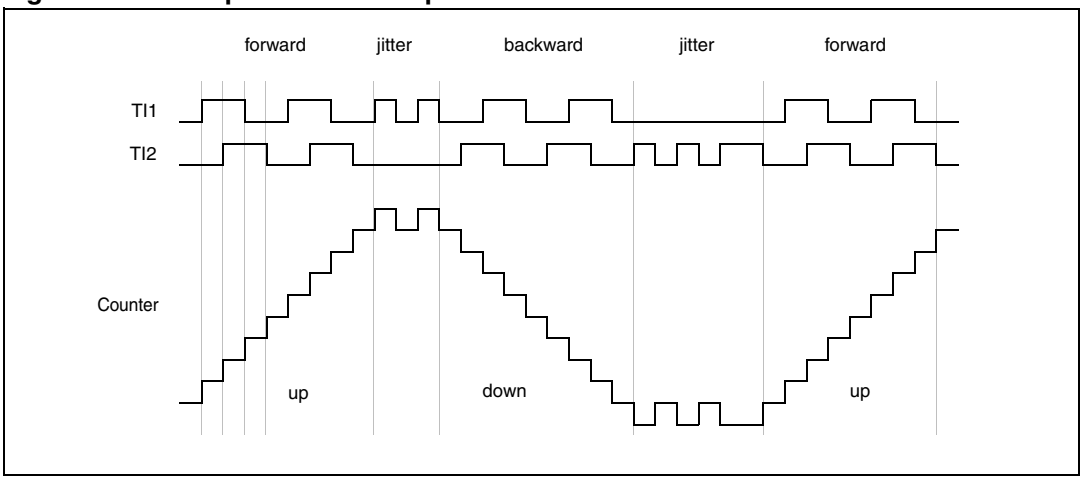
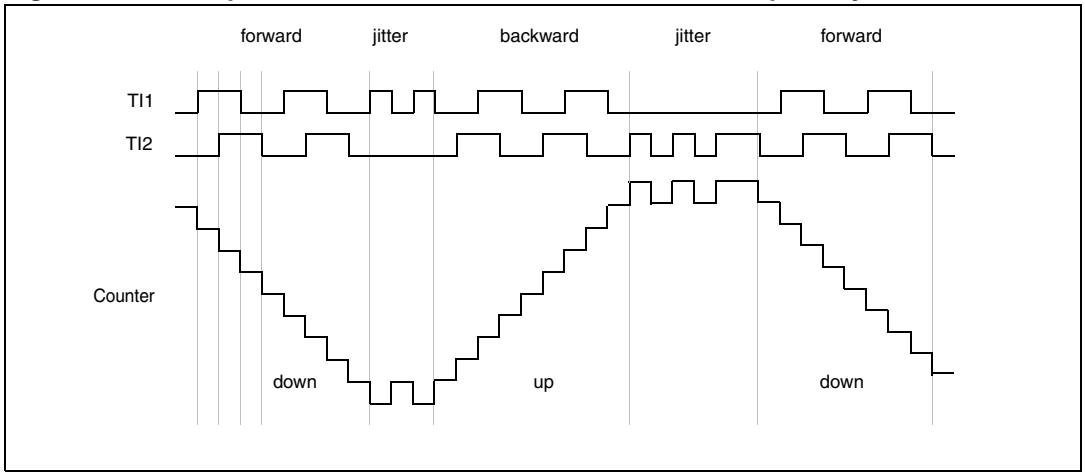


Figure 123 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 123. 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. You can obtain dynamic information (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. You can do this 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.

13.3.13 Timer input XOR function

The TI1S bit in the TIM1_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 12.3.18 on page 242*.

13.3.14 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:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture

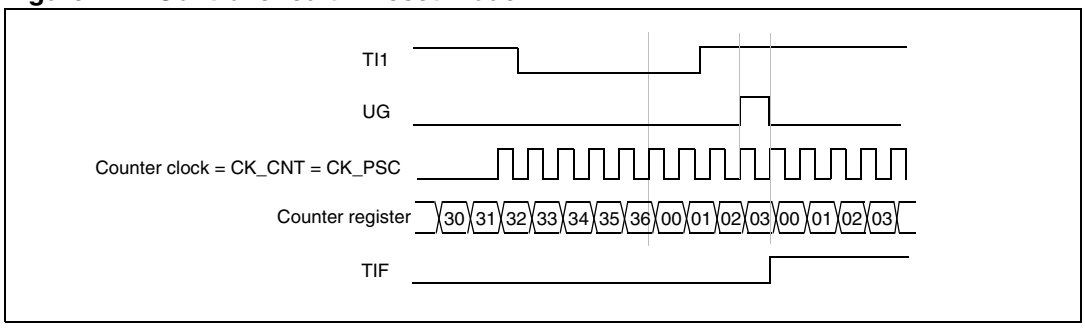
prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).

- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 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 124. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

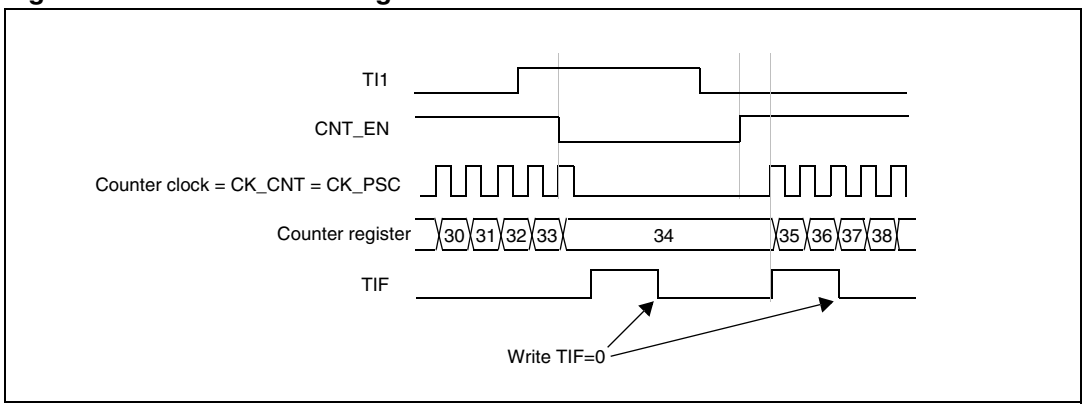
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. 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=101 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 125. Control circuit in gated mode



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.

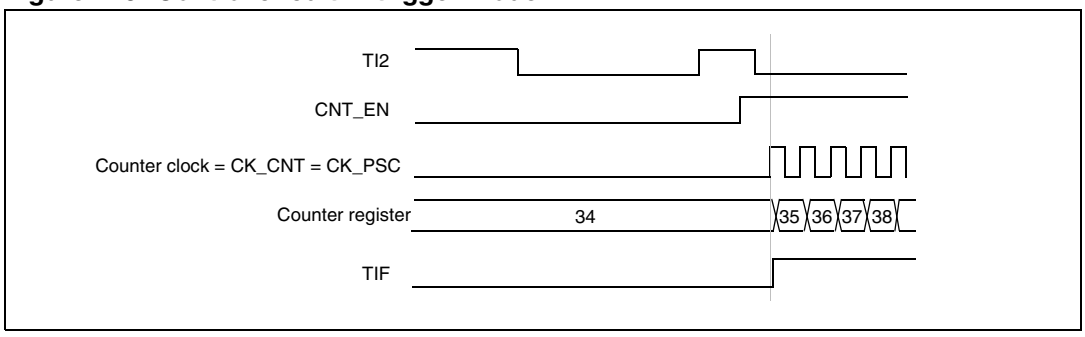
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. 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).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 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 126. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode,

gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

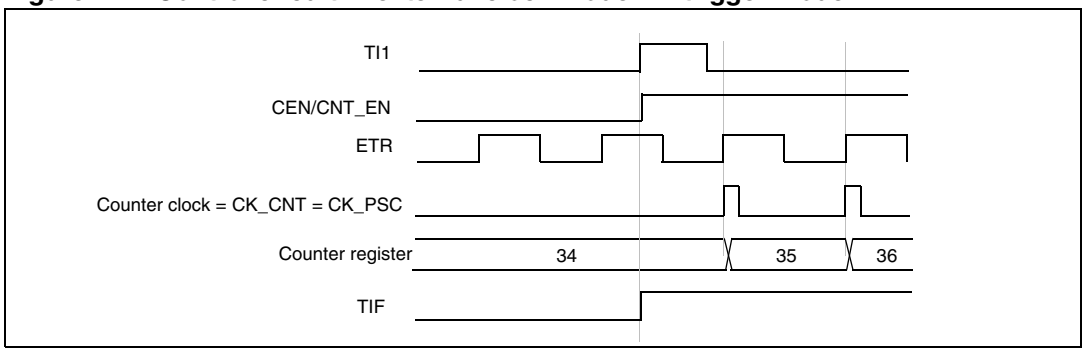
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in 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=101 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 127. Control circuit in external clock mode 2 + trigger mode



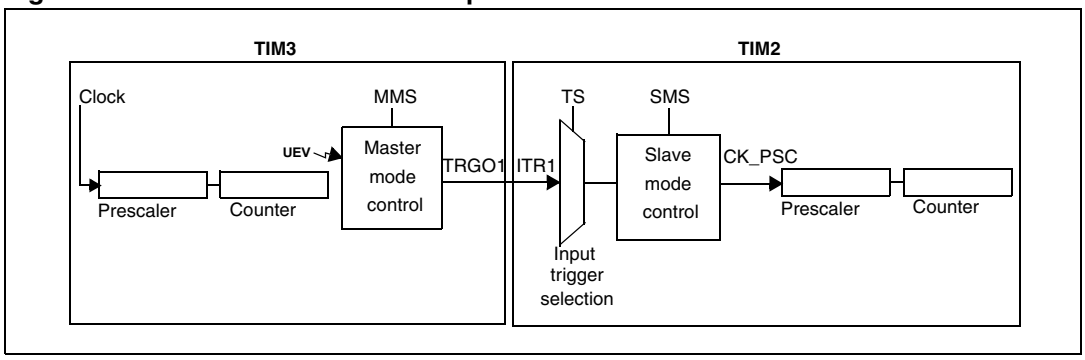
13.3.15 Timer synchronization

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 128: Master/Slave timer example presents an overview of the trigger selection and the master mode selection blocks.

Using one timer as prescaler for another

Figure 128. Master/Slave timer example



For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to *Figure 128*. To do this:

- Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM3_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
- To connect the TRGO1 output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR1 as internal trigger. You select this through the TS bits in the TIM2_SMCR register (writing TS=000).
- Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
- Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

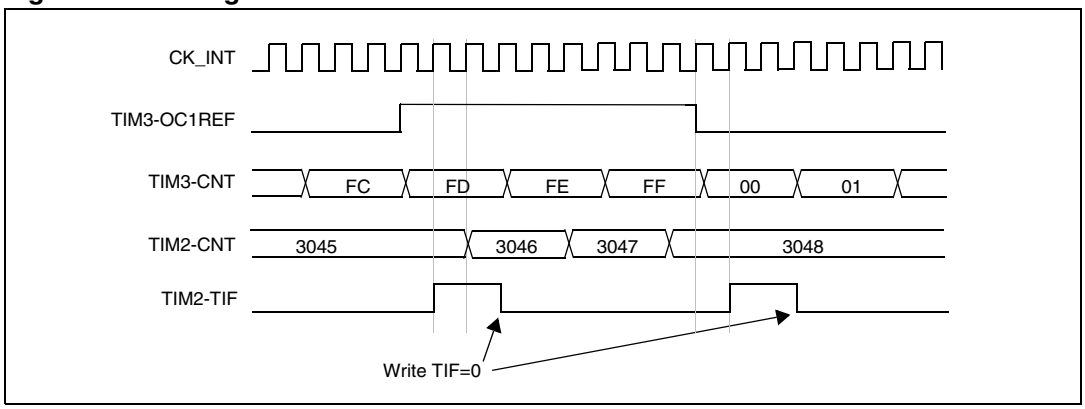
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 1. Refer to *Figure 128* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

- Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
- Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
- Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 129. Gating TIM2 with OC1REF of TIM3

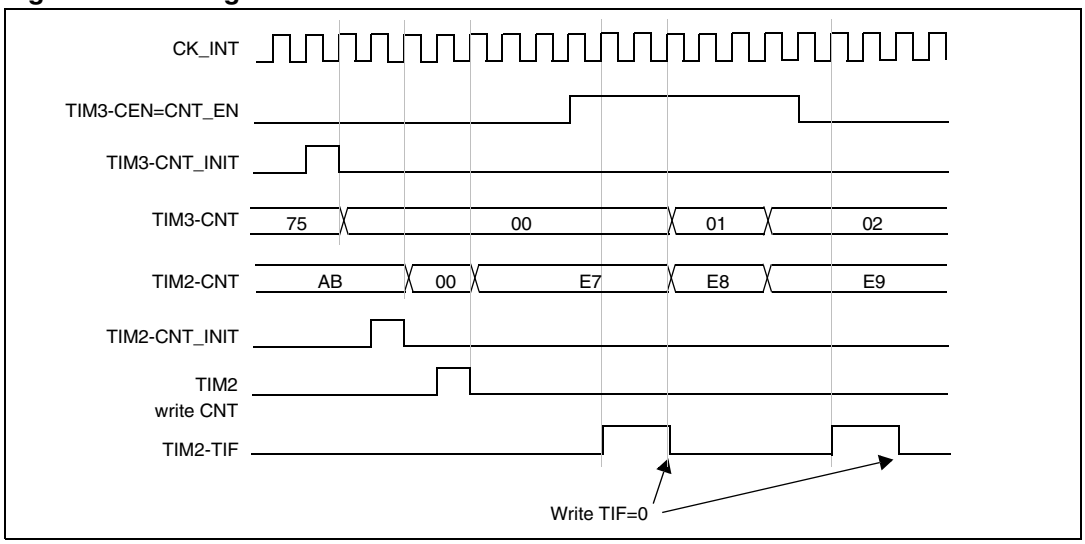


In the example in *Figure 129*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. You can then write any value you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example, we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing ‘0 to the CEN bit in the TIM3_CR1 register:

- Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
- Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
- Reset TIM3 by writing ‘1 in UG bit (TIM3_EGR register).
- Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
- Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
- Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).
- Stop TIM3 by writing ‘0 in the CEN bit (TIM3_CR1 register).

Figure 130. Gating TIM2 with Enable of TIM3

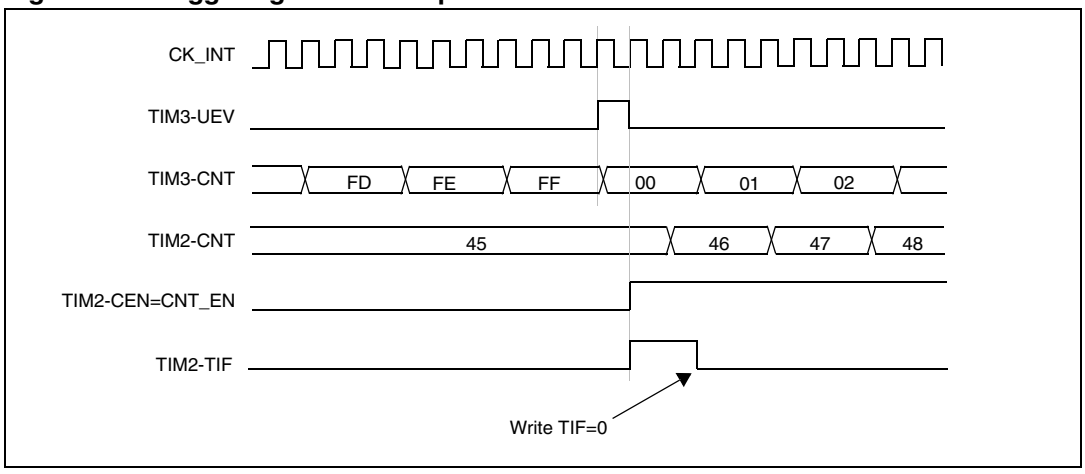


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 1. Refer to *Figure 128* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

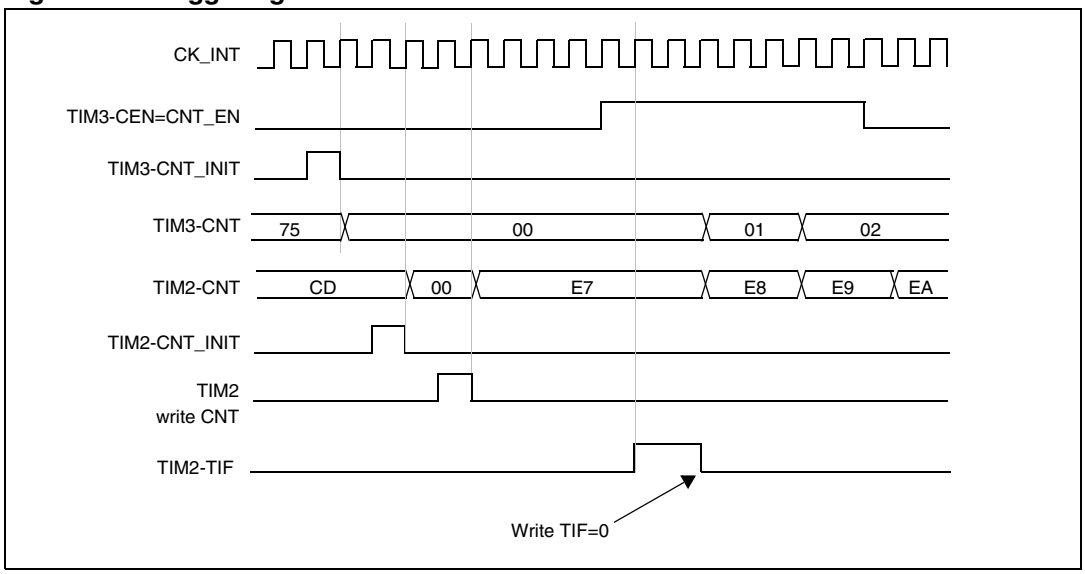
- Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
- Configure the TIM3 period (TIM3_ARR registers).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
- Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Figure 131. Triggering TIM2 with update of TIM3



As in the previous example, you can initialize both counters before starting counting. *Figure 132* shows the behavior with the same configuration as in *Figure 131* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 132. Triggering TIM2 with Enable of TIM3



Using one timer as prescaler for another timer

For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to *Figure 128* for connections. To do this:

- Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register). then it outputs a periodic signal on each counter overflow.
- Configure the TIM3 period (TIM1_ARR registers).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in external clock mode 1 (SMS=111 in TIM2_SMCR register).
- Start TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start TIM1 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Starting 2 timers synchronously in response to an external trigger

In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to *Figure 128* for connections. To ensure the counters are aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

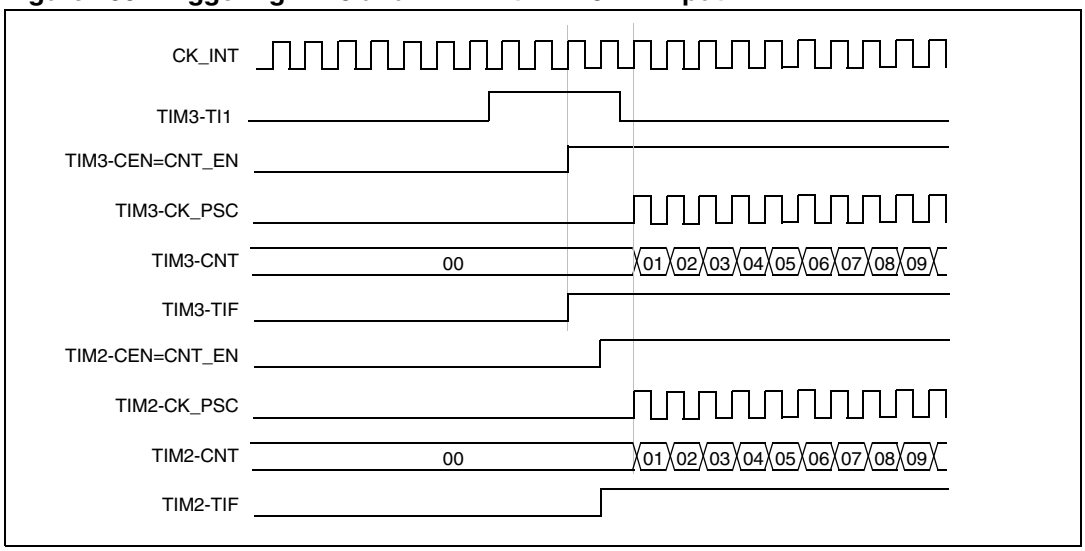
- Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
- Configure TIM1 slave mode to get the input trigger from TI1 (TS=100 in the TIM3_SMCR register).
- Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
- Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).

- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx_CNT). You can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.

Figure 133. Triggering TIM3 and TIM2 with TIM3 TI1 input



13.3.16 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 core - halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to *Section 25.15.2: Debug support for timers, watchdog and I2C*.

13.4 TIMx2 to TIM5 registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

13.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, always read as 0

Bits 9:8 **CKD**: 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: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
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**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

13.4.2 TIMx control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
See also *Section 12.3.18: Interfacing with Hall sensors on page 242*

Bits 6:4 **MMS**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)
100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, always read as 0

13.4.3 TIMx slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge
1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).
2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).
3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.
00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N 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

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 6:4 **TS:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0).
001: Internal Trigger 1 (ITR1).
010: Internal Trigger 2 (ITR2).
011: Internal Trigger 3 (ITR3).
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)
111: External Trigger input (ETRF)
See *Table 71: TIMx internal trigger connection on page 315* 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 2:0 **SMS:** 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.
000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.
001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
101: 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.
110: 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.
111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 71. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	TIM15	TIM3	TIM4
TIM3	TIM1	TIM2	TIM15	TIM4
TIM4	TIM1	TIM2	TIM3	TIM15

13.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 15 Reserved, always read as 0.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 Reserved, always read as 0

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, always read as 0.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 Reserved, always read as 0.

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

13.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 15:13 Reserved, always read as 0.

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, always read as 0.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 Reserved, always read as 0

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.

0: No match

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

At overflow or underflow (for TIM2 to TIM4) and if UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

13.4.6 TIMx event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
refer to CC1G description

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).

13.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). 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. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: 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

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 6:4 **OC1M**: 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.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: 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).
010: 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).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: 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).
111: 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.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).*

***2:** In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=00 (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently 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**: 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).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: 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**: 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 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

Note: In current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: 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**: 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).

13.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC4CE:** Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: 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 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: 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).

Input capture mode

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: 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**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: 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).

13.4.9 TIMx capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

Bits 15:14 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

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/Compare 3 output polarity
refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
refer to CC1E description

Bits 7:6 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

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
CC1 channel configured as output:
0: OC1 active high.
1: OC1 active low.
CC1 channel configured as input:
This bit selects whether IC1 or IC1 is used for trigger or capture operations.
0: non-inverted: capture is done on a rising edge of IC1. When used as external trigger, IC1 is non-inverted.
1: inverted: capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted.

Bit 0 **CC1E**: Capture/Compare 1 output enable
CC1 channel configured as output:
0: Off - OC1 is not active.
1: On - OC1 signal is output on the corresponding output pin.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Table 72. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Polarity, OCx_EN=1

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

13.4.10 TIMx counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

13.4.11 TIMx prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

13.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 13.3.1: Time-base unit on page 276* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

13.4.13 TIMx capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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 CC1is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

13.4.14 TIMx capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 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 signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

13.4.15 TIMx capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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_CCMR3 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 CC3is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (IC3).

13.4.16 TIMx capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

1/ if CC4 channel is configured as output (CC4S bits):

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

2/ if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):

CCR4 is the counter value transferred by the last input capture 4 event (IC4).

13.4.17 TIMx DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, always read as 0

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, always read as 0

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..

13.4.18 TIMx DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
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).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) 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

Note: 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.

13.4.19 TIMx register map

TIMx registers are mapped as described in the table below:

Table 73. TIMx register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Reserved																								TI1S	MMS[2:0]			CCDS	Reserved		
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Reserved																ETP	ECE	ETP S [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Reserved																									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0

Table 73. TIMx 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
0x18	**TIMx_CCMR1** *Output Compare mode*	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 *Input Capture mode*	Reserved																IC2F[3:0]				IC2PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** *Output Compare mode*	Reserved																O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 *Input Capture mode*	Reserved																IC4F[3:0]				IC4PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Reserved																		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
	Reset value																			0	0			0	0			0	0			0	0
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIMx_CCR3**	Reserved																CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIMx_CCR4**	Reserved																CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																

Table 73. TIMx 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
0x48	**TIMx_DCR**	Reserved																			DBL[4:0]					Reserved			DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIMx_DMAR**	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

14 General-purpose timers (TIM12/13/14)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 Kbytes and 1 Mbyte.

This section applies to high density value line devices only.

14.1 TIM12/13/14 introduction

The TIM12/13/14 general-purpose 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).

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 TIM12/13/14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 14.4.12*.

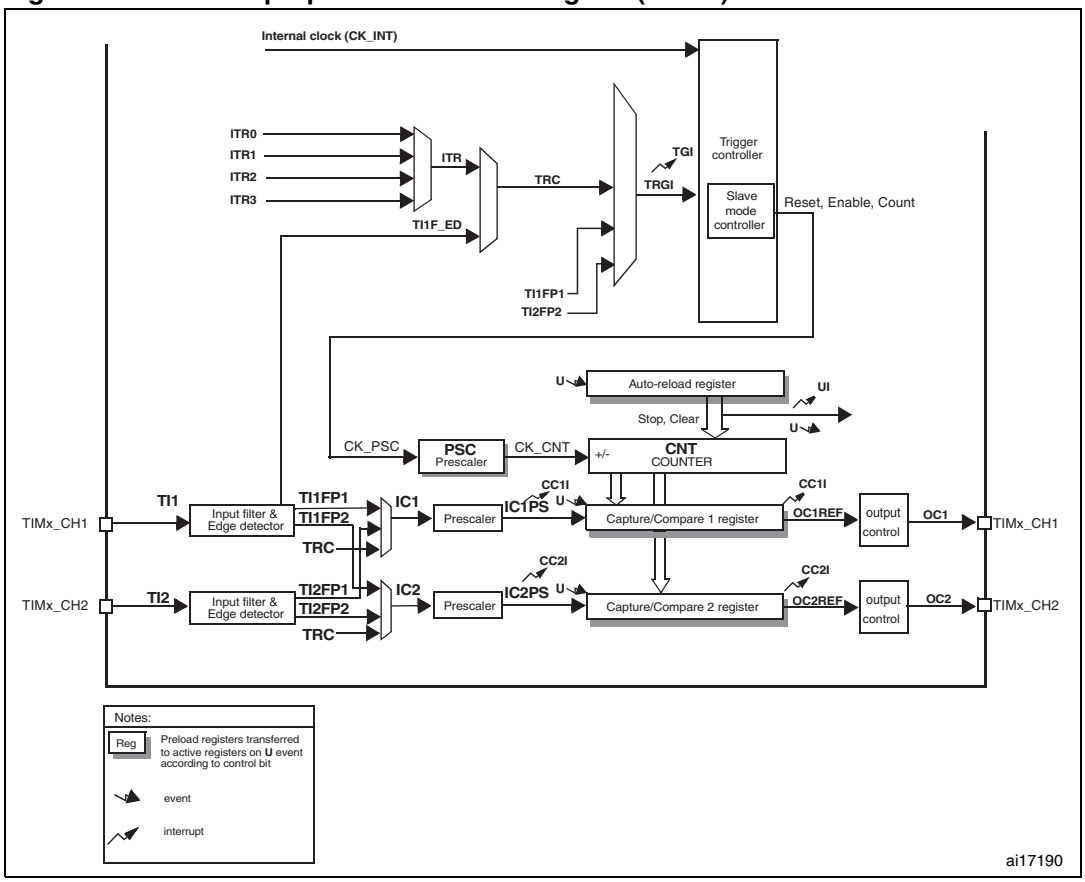
14.2 TIM12/13/14 main features

14.2.1 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65535 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare

Figure 134. General-purpose timer block diagram (TIM12)

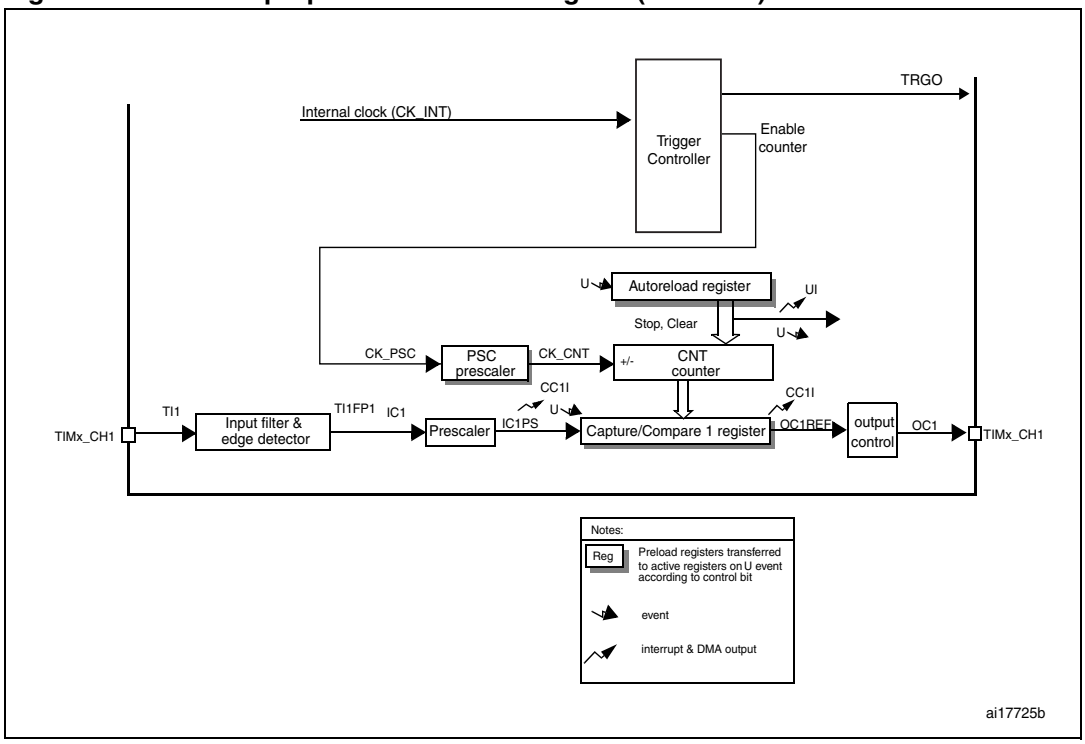


14.3 TIM13/TIM14 main features

The features of general-purpose timers TIM13/TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65535 (can be changed “on the fly”)
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

Figure 135. General-purpose timer block diagram (TIM13/14)



14.4 TIM12/13/14 functional description

14.4.1 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. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in 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 137 and *Figure 138* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 136. Counter timing diagram with prescaler division change from 1 to 2

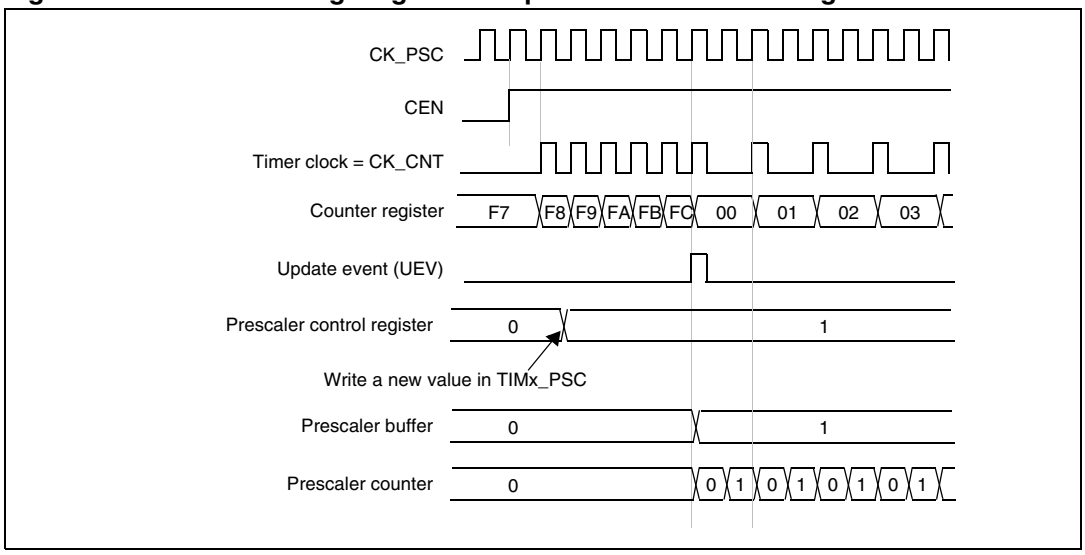
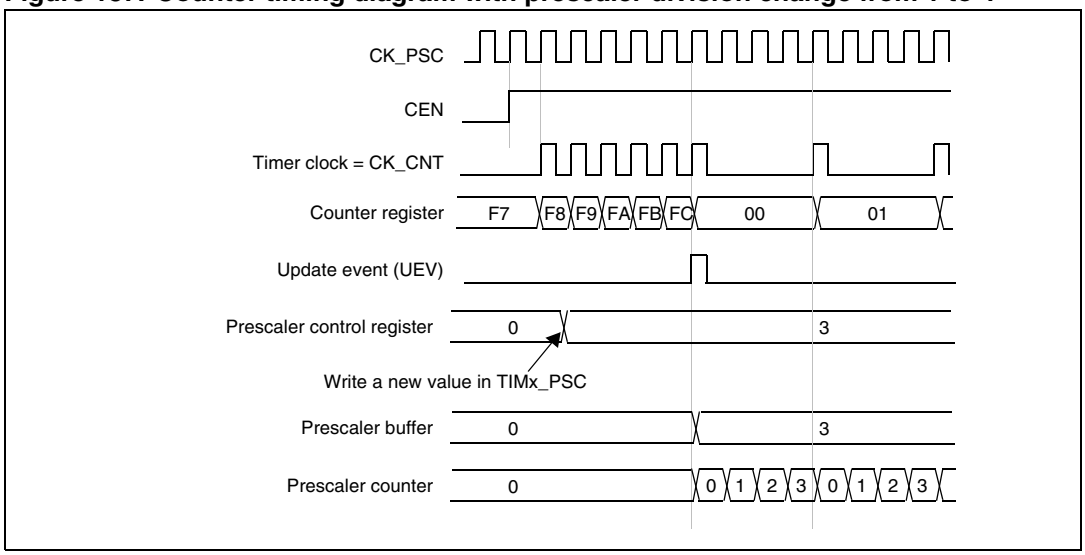


Figure 137. Counter timing diagram with prescaler division change from 1 to 4



14.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without

setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 138. Counter timing diagram, internal clock divided by 1

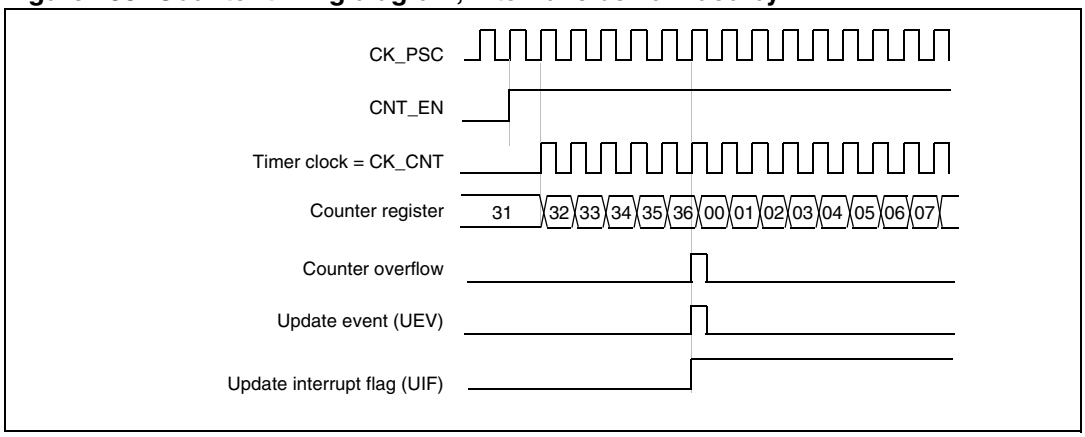


Figure 139. Counter timing diagram, internal clock divided by 2

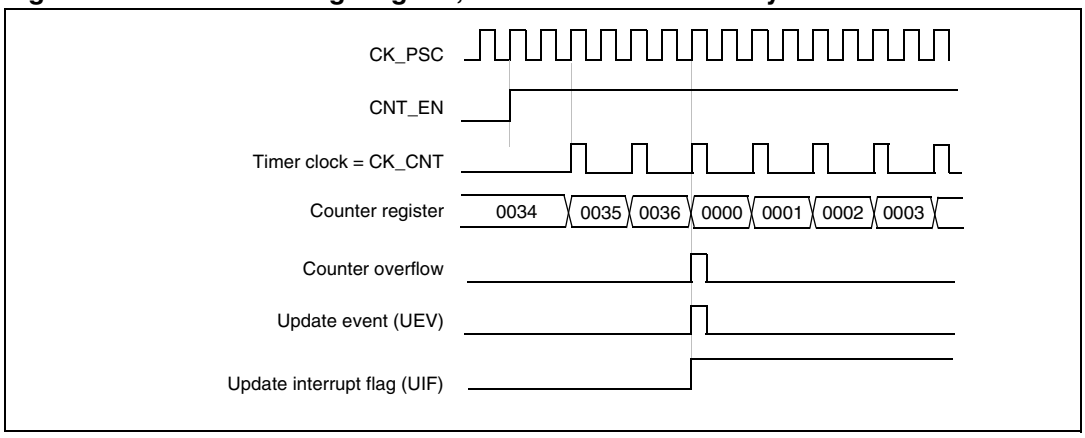


Figure 140. Counter timing diagram, internal clock divided by 4

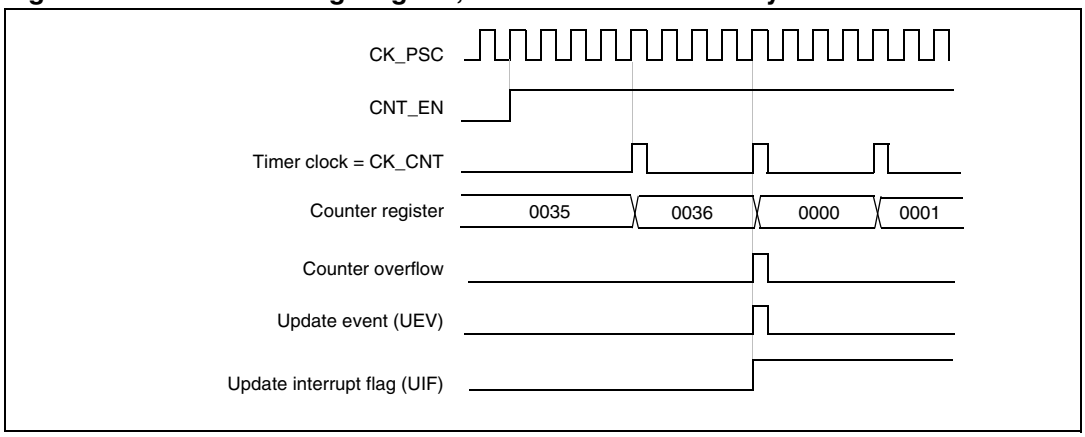


Figure 141. Counter timing diagram, internal clock divided by N

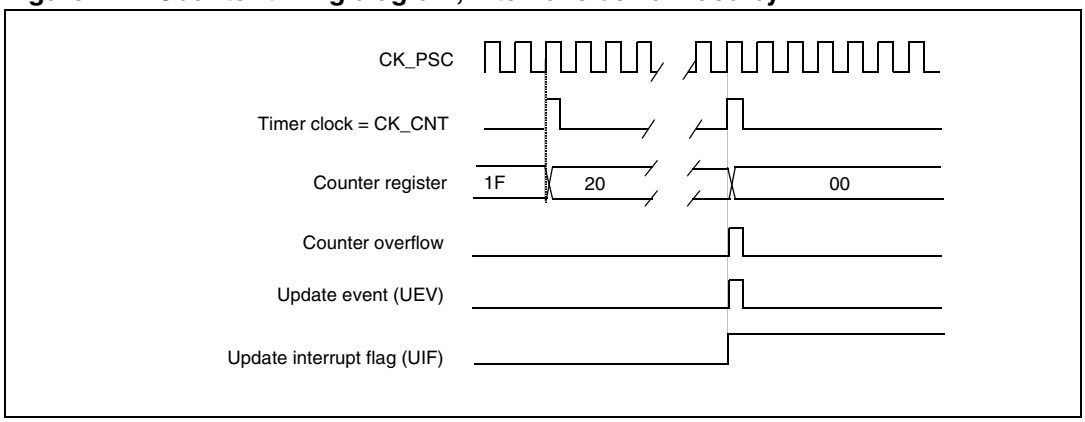


Figure 142. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

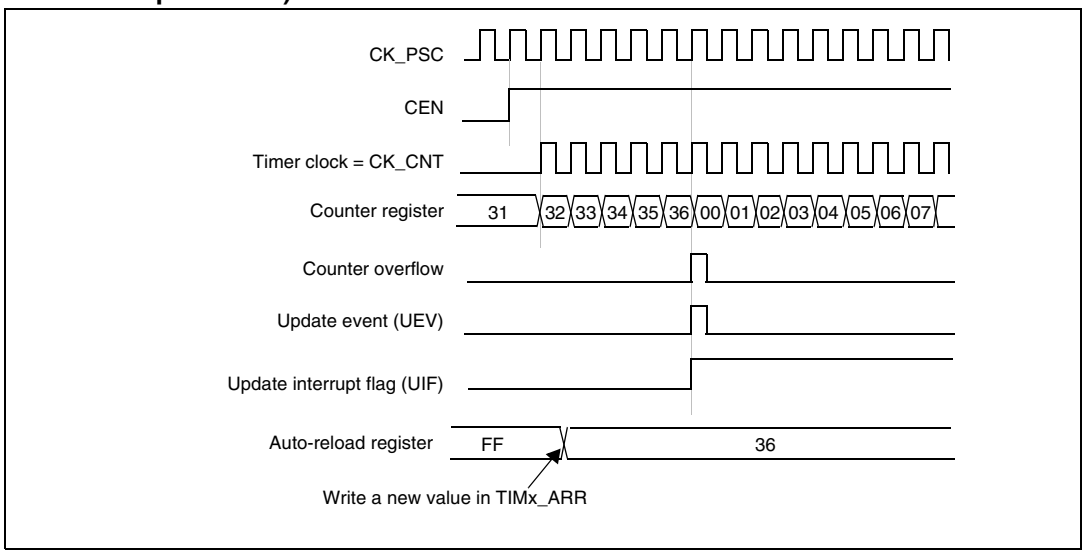
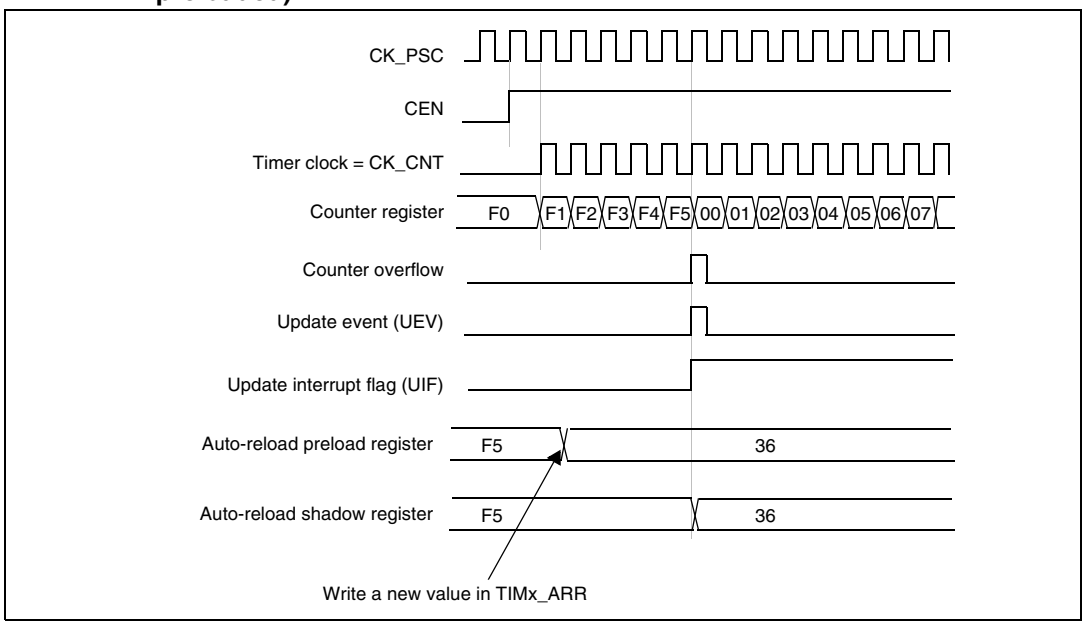


Figure 143. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



14.4.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1 (for **TIM12**): external input pin (TIx)
- Internal trigger inputs (ITRx) (for **TIM12**): connecting the trigger output from another timer. Refer to *Using one timer as prescaler for another* for more details.

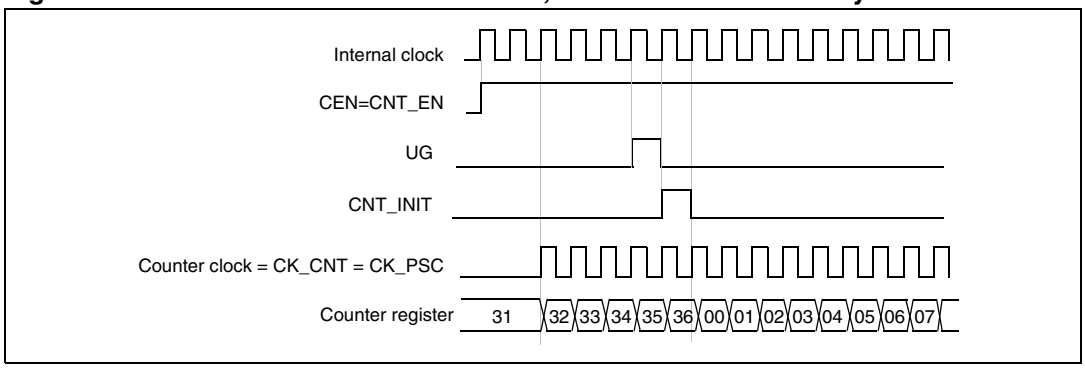
Internal clock source (CK_INT)

The internal clock source is the default clock source for TIM13/TIM14.

For TIM9 and TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS='000'). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 144 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

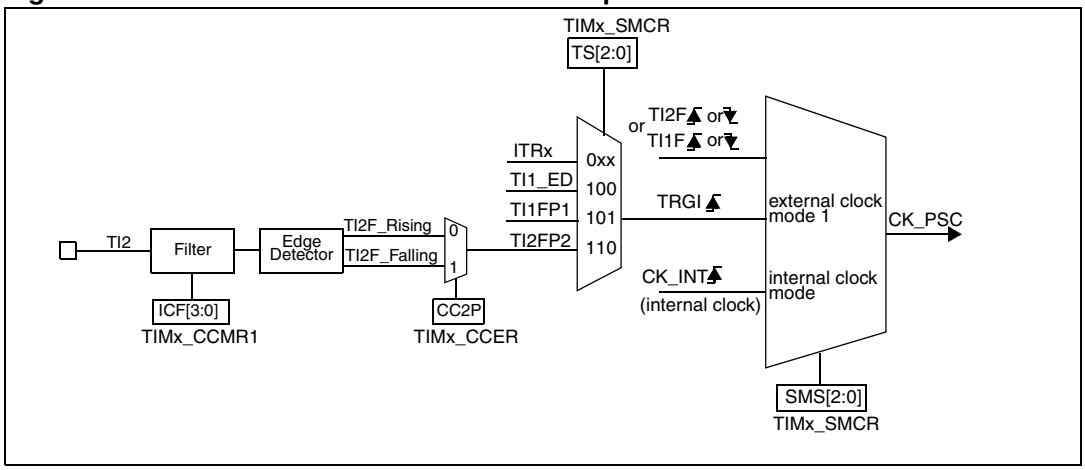
Figure 144. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1(TIM12)

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 145. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

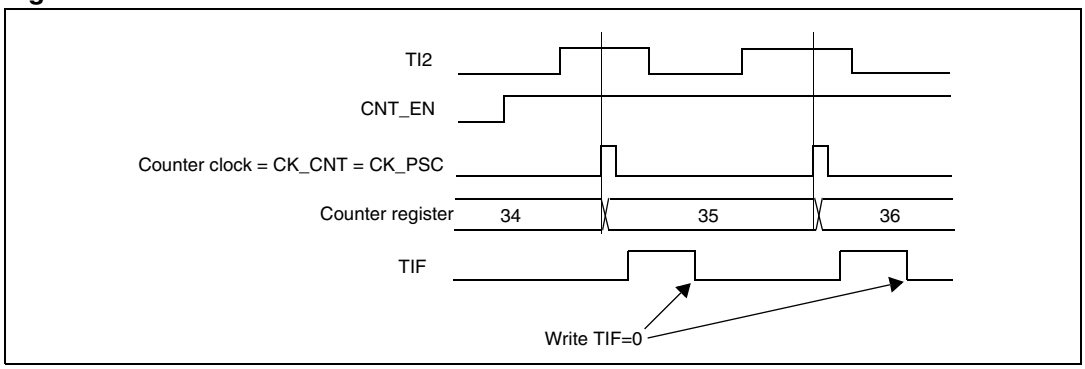
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F='0000').
3. Select the rising edge polarity by writing CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS='111' in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS='110' in the TIMx_SMCR register.
6. Enable the counter by writing CEN='1' in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don't 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.

Figure 146. Control circuit in external clock mode 1



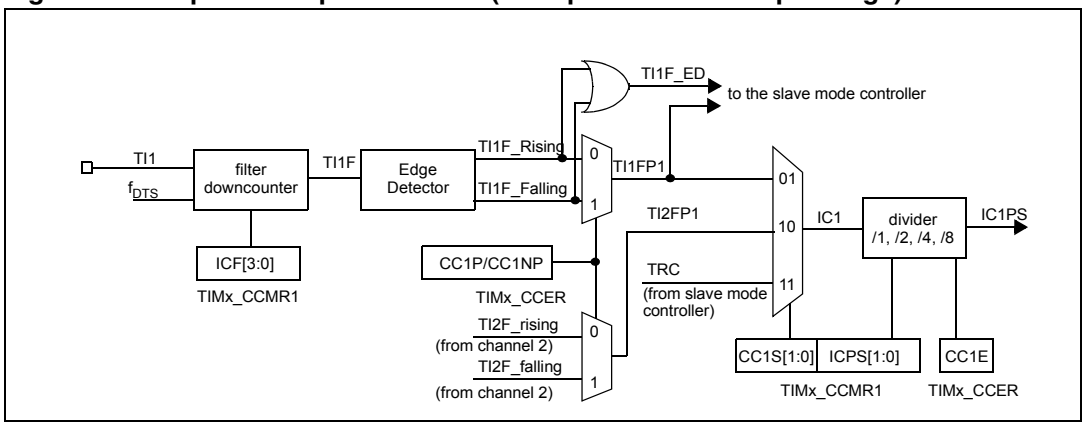
14.4.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 147 to *Figure 149* 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 147. 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 148. Capture/compare channel 1 main circuit

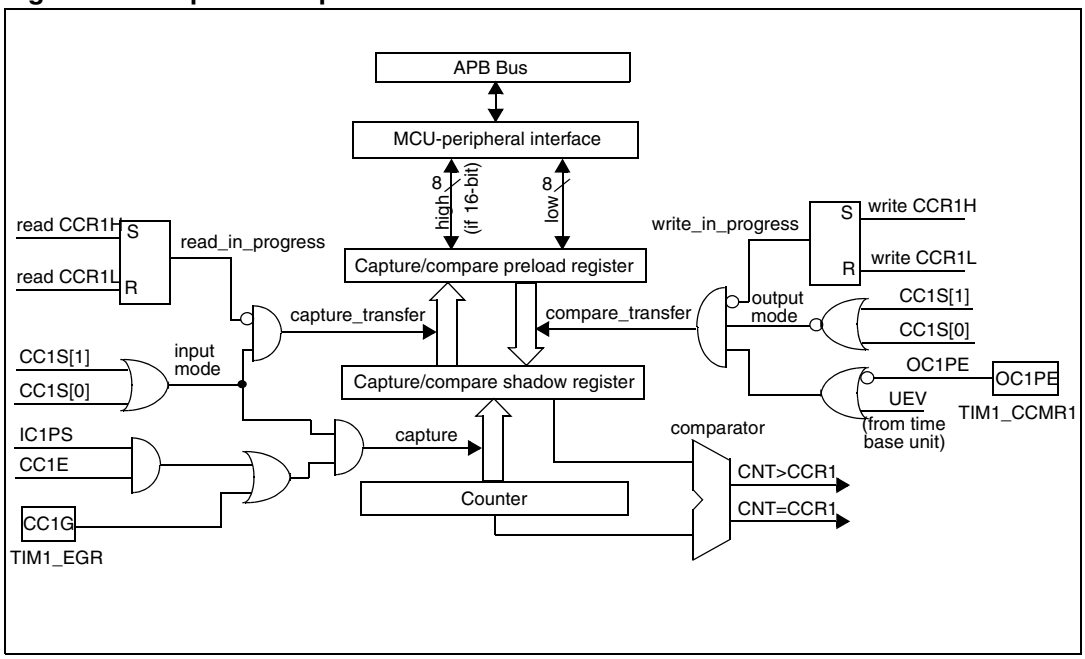
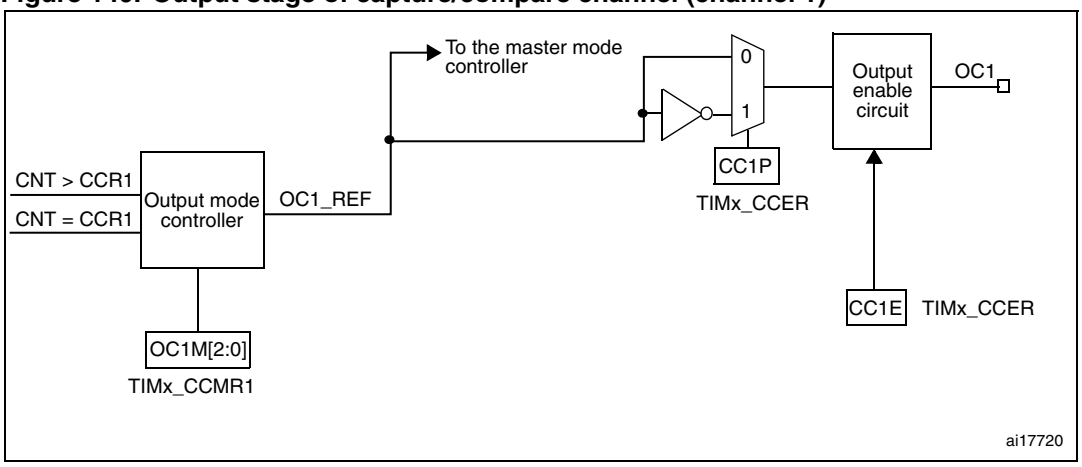


Figure 149. Output stage of capture/compare channel (channel 1)



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.

14.4.5 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 you write it to '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 active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to '01' in the TIMx_CCMR1 register. As soon as CC1S becomes different from '00', the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect 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 f_{DTS} frequency). Then write IC1F bits to '0011' in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to '00' in the TIMx_CCER register (rising edge in this case).
4. 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).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE 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.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

14.4.6 PWM input mode (only for TIM12)

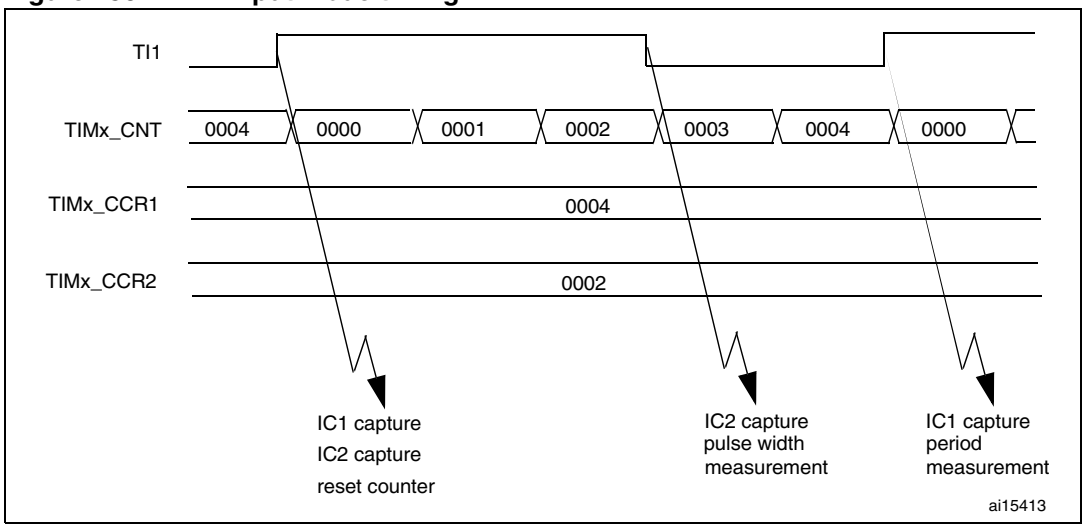
This mode is a particular case of input capture mode. The procedure is the same except:

- 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, you 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 active input for TIMx_CCR1: write the CC1S bits to '01' in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to '00' (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to '10' in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to '11' (active on falling edge).
5. Select the valid trigger input: write the TS bits to '101' in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to '100' in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 150. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

14.4.7 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, you just need 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 requests can be sent accordingly. This is described in the output compare mode section below.

14.4.8 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:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM='000'), be set active (OCxM='001'), be set inactive (OCxM='010') or can toggle (OCxM='011') on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

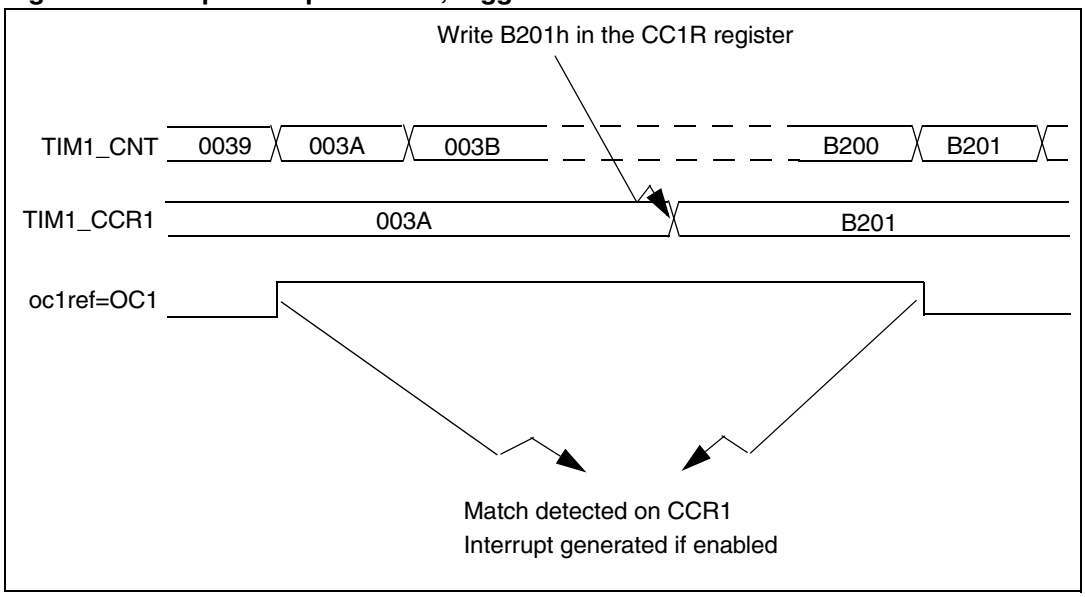
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = '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 151*.

Figure 151. Output compare mode, toggle on OC1.



14.4.9 PWM mode

Pulse Width Modulation mode allows you 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. You must enable the corresponding preload register 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, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

The OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

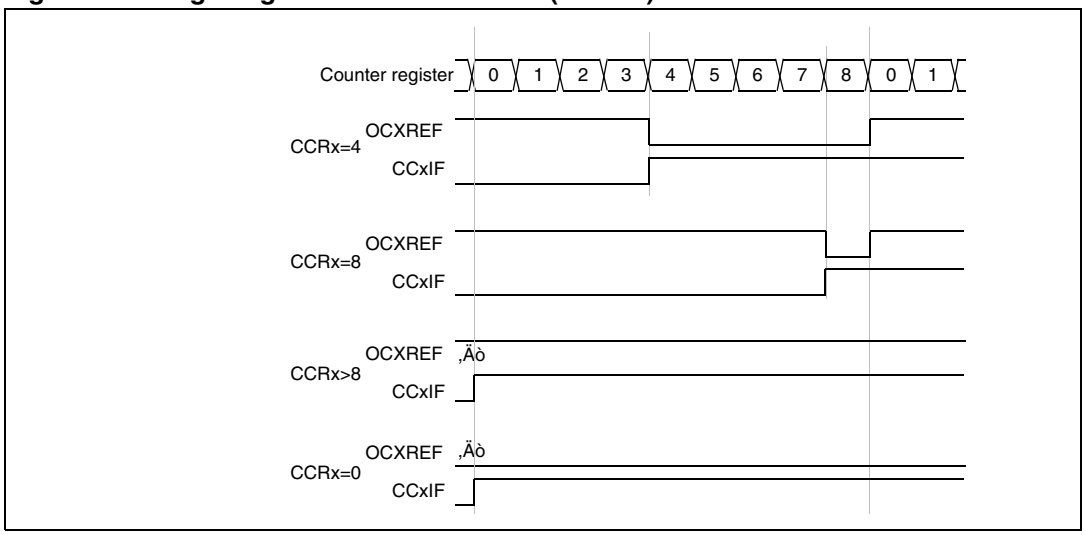
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

PWM edge-aligned mode

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 152* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 152. Edge-aligned PWM waveforms (ARR=8)



14.4.10 One-pulse mode (only for TIM12)

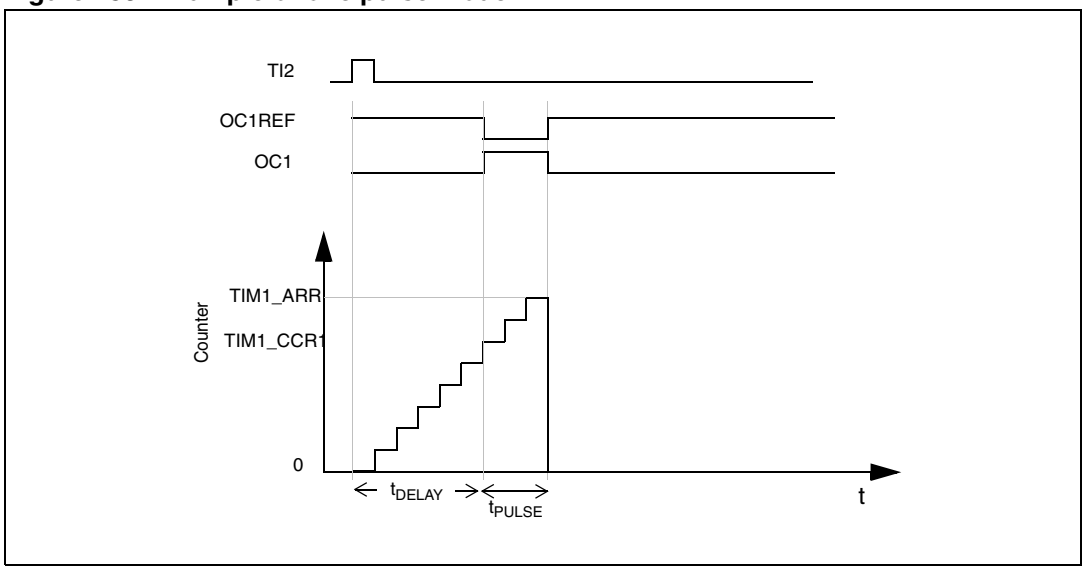
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx≤ ARR (in particular, 0 < CCRx)

Figure 153. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP = '0' in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
4. 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_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you 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 you enable PWM mode 2 by writing OC1M='111' in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have 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.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit 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.

14.4.11 TIM12 external trigger synchronization

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

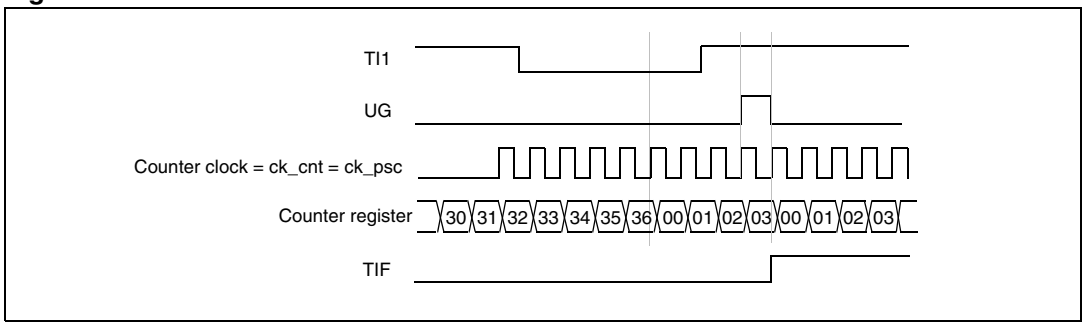
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = '01' in the TIMx_CCMR1 register. Program CC1P and CC1NP to '00' in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS='100' in TIMx_SMCR register. Select TI1 as the input source by writing TS='101' 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 154. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

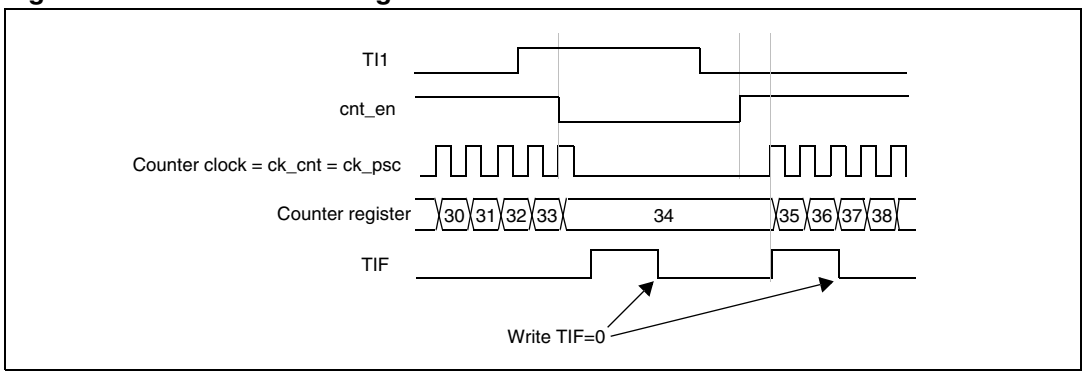
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S='01' in TIMx_CCMR1 register. Program CC1P='1' and CC1NP= '0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS='101' in TIMx_SMCR register. Select TI1 as the input source by writing TS='101' 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 155. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

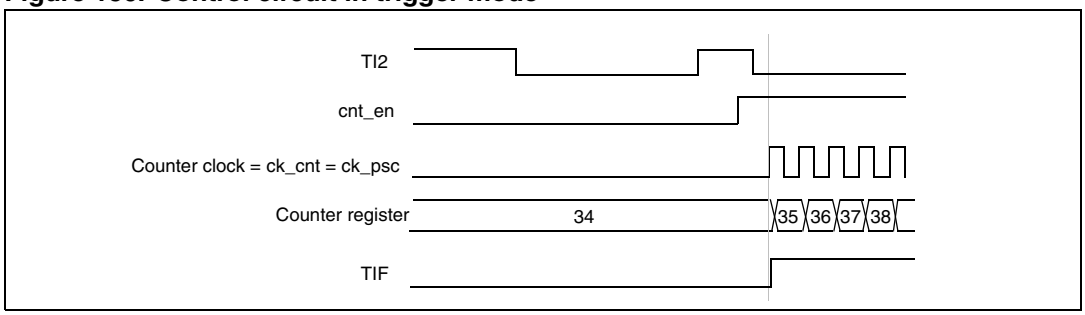
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F='0000'). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S='01' in TIMx_CCMR1 register. Program 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='110' 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 156. Control circuit in trigger mode



14.4.12 Timer synchronization (TIM12)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 13.3.15: Timer synchronization on page 305* for details.

14.4.13 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 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 25.15.2: Debug support for timers, watchdog and I2C*.

14.5 TIM12 registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

14.5.1 TIM12 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

Bits 15:10 Reserved, always read as 0

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled:
- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
- Counter overflow
- Setting the UG bit

Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

14.5.2 TIM12 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									MMS[2:0]			Reserved			
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in Master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit in the TIMx_EGR register is used as the 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 the trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between 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 TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).
010: **Update** - The update event is selected as the trigger output (TRGO). For instance a master timer can be used as a prescaler for a slave timer.
011: **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 a compare match occurs. (TRGO).
100: **Compare** - OC1REF signal is used as the trigger output (TRGO).
101: **Compare** - OC2REF signal is used as the trigger output (TRGO).
110: Reserved
111: Reserved

Bits 3:0 Reserved, must be kept at reset value.

14.5.3 TIM12 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

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

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 in order to synchronize several timers on a single external event.

Bits 6:4 **TS:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

000: Internal Trigger 0 (ITR0)

001: Internal Trigger 1 (ITR1)

010: Internal Trigger 2 (ITR2)

011: Internal Trigger 3 (ITR3)

100: TI1 Edge Detector (TI1F_ED)

101: Filtered Timer Input 1 (TI1FP1)

110: Filtered Timer Input 2 (TI2FP2)

111: Reserved.

See *Table 74: TIMx Internal trigger connection on page 356* for more details on the meaning of ITRx 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 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** 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 descriptions.

000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock

001: Reserved

010: Reserved

011: Reserved

100: Reset mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers

101: 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. Counter starts and stops are both controlled

110: Trigger mode - The counter starts on a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled

111: External clock mode 1 - Rising edges of the selected trigger (TRGI) clock the counter

Note: The Gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the Gated mode checks the level of the trigger signal.

Table 74. TIMx Internal trigger connection(1)

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM12	TIM4	TIM5	TIM13	TIM14

1. When a timer is not present in the product, the corresponding trigger ITRx is not available.

14.5.4 TIM12 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIE	Res			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bit 15:7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5:3 Reserved, must be kept at reset value.

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.

14.5.5 TIM12 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

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

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5:3 Reserved, always read as 0

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–At overflow and if UDIS='0' in the TIMx_CR1 register.

–When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.

–When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS='0' and UDIS='0' in the TIMx_CR1 register.

14.5.6 TIM12 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved			CC2G	CC1G	UG
									w				w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software 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:

the CC1IF flag is set, the corresponding interrupt is sent if enabled.

If channel CC1 is configured as input:

The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt 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-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

14.5.7 TIM12 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes. For a given bit, OCxx describes its function when the channel is configured in output mode, ICxx describes its function when the channel is configured in input mode. So you must take care that the same bit can have different meanings for the input stage and the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	IC2F[3:0]			IC2PSC[1:0]					IC1F[3:0]			IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 Reserved

Bits 14:12 **OC2M[2:0]**: 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 bitfield 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 works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved

Bits 6:4 **OC1M**: 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 the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits, respectively.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: Set channel 1 to active level on match. The OC1REF signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
010: Set channel 1 to inactive level on match. The OC1REF signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1
100: Force inactive level - OC1REF is forced low
101: Force active level - OC1REF is forced high
110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1, else it is active (OC1REF=’1’)
111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else it is inactive.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in the TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the 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 the 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**: Capture/Compare 1 selection

This bitfield 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 works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bitfield 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 works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bitfield defines the frequency used to sample the TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, 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

Note: In the current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the 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**: Capture/Compare 1 selection

This bitfield 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: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

14.5.8 TIM12 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

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

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
refer to CC1NP description

Bits 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (refer to CC1P description).

Bits 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
CC1 channel configured as output:
0: OC1 active high.
1: OC1 active low.
CC1 channel configured as input:
CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.
Note: 11: noninverted/both edges
Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
CC1 channel configured as output:
0: Off - OC1 is not active.
1: On - OC1 signal is output on the corresponding output pin.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Table 75. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (OCx='0', OCx_EN='0')
1	OCx=OCxREF + Polarity, OCx_EN='1'

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and on the GPIO registers.

14.5.9 TIM12 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

14.5.10 TIM12 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event.

14.5.11 TIM12 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to the *Section 14.4.1: Time-base unit on page 336* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

14.5.12 TIM12 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded into the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit). Else the preload value is copied into the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signaled on the OC1 output.

If channel CC1is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

14.5.13 TIM12 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded into the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit). Else the preload value is copied into the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signalled on the OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

14.5.14 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described in the table below:

Table 76. TIM12 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	Reserved	Reserved	Reserved	OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x04	**TIMx_CR2**	Reserved																									MMS[2:0]			Reserved			
	Reset value																										0	0	0				

Table 76. TIM12 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	**TIMx_SMCR**	Reserved																								MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																									0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																									TIE	Reserved	Reserved	Reserved	CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0
0x10	**TIMx_SR**	Reserved																					CC2OF	CC1OF	Reserved	Reserved	TIF	Reserved	Reserved	Reserved	CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x14	**TIMx_EGR**	Reserved																									TG	Reserved	Reserved	Reserved	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x18	**TIMx_CCMR1** *Output Compare mode*	Reserved																	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Reserved	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 *Input Capture mode*	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved																																
0x20	**TIMx_CCER**	Reserved																								CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x4C	Reserved																																

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

14.6 TIM13/14 registers

14.6.1 TIM13/14 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved				URS	UDIS	CEN
						rw	rw	rw					rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: 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: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:
- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:
- Counter overflow
- Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

14.6.2 TIM13/14 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

14.6.3 TIM13/14 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Reserved							CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bit 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value. It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.
If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred.
1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow and if UDIS='0' in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.

14.6.4 TIM13/14 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
														w	w

Bits 15:2 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 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 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.

14.6.5 TIM13/14 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). 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. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
Reserved								IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 15:7 Reserved

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

001: 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).

010: 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).

011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active.

Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:8 Reserved

Bits 7:4 **IC1F**: 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 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

Note: In current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: 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**: 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:
11:

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

14.6.6 TIM13/14 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.
CC1 channel configured as output: CC1NP must be kept cleared.
CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
The CC1P bit selects TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TI1FP1 rising edge (capture mode), TI1FP1 is not inverted.
01: inverted/falling edge
Circuit is sensitive to TI1FP1 falling edge (capture mode), TI1FP1 is inverted.
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TI1FP1 rising and falling edges (capture mode), TI1FP1 is not inverted.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
CC1 channel configured as output:
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 77. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx='0', OCx_EN='0')
1	OCx=OCxREF + Polarity, OCx_EN='1'

Note: The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO registers.

14.6.7 TIM13/14 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

14.6.8 TIM13/14 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

14.6.9 TIM13/14 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 14.4.1: Time-base unit on page 336* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

14.6.10 TIM13/14 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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 CC1is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

14.6.11 TIM13/14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 78. TIM13/14 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	Reserved	Reserved	Reserved	Reserved	URS	UDIS	CEN
	Reset value																							0	0	0					0	0	0
0x08	**TIMx_SMCR**	Not Available																															
	Reset value																																
0x0C	**TIMx_DIER**	Reserved																														CC1IE	UIE
	Reset value																															0	0
0x10	**TIMx_SR**	Reserved																						CC1OF	Reserved							CC1IF	UIF
	Reset value																							0								0	0
0x14	**TIMx_EGR**	Reserved																														CC1G	UG
	Reset value																															0	0
0x18	**TIMx_CCMR1** *Output compare mode*	Reserved																									OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 *Input capture mode*	Reserved																								IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x1C	Reserved																																
0x20	**TIMx_CCER**	Reserved																						Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CC1NP	Reserved	CC1P	CC1E
	Reset value																													0		0	0

Table 78. TIM13/14 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
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	**TIMx_CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x4C	Reserved																																

15 General-purpose timers (TIM15/16/17)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

15.1 TIM15/16/17 introduction

The TIM15/16/17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/16/17 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 14.4.12*.

15.2 TIM15 main features

TIM15 includes the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divid (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state

- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

15.3 TIM16 and TIM17 main features

The TIM16 and 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
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input

Figure 157. TIM15 block diagram

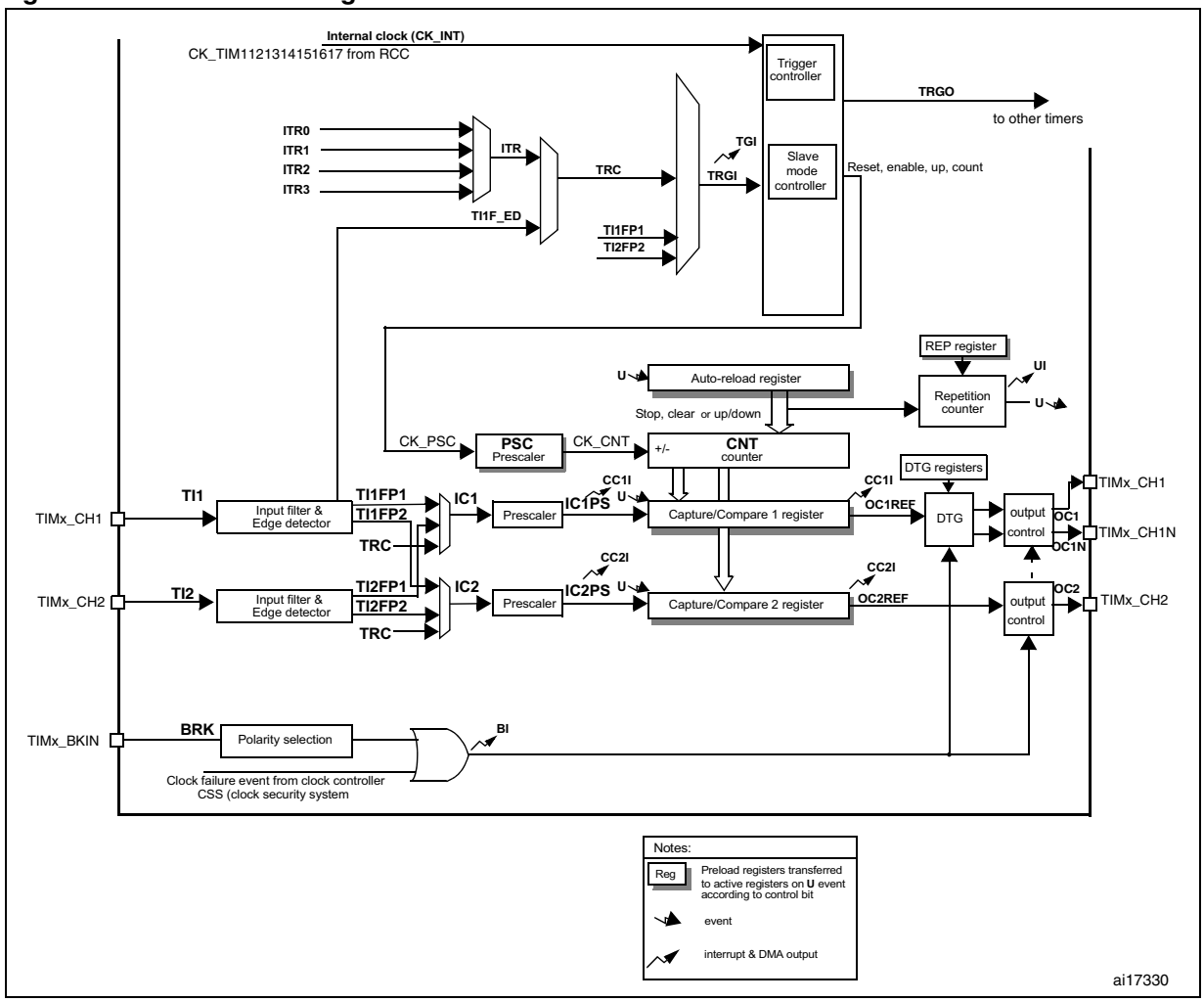
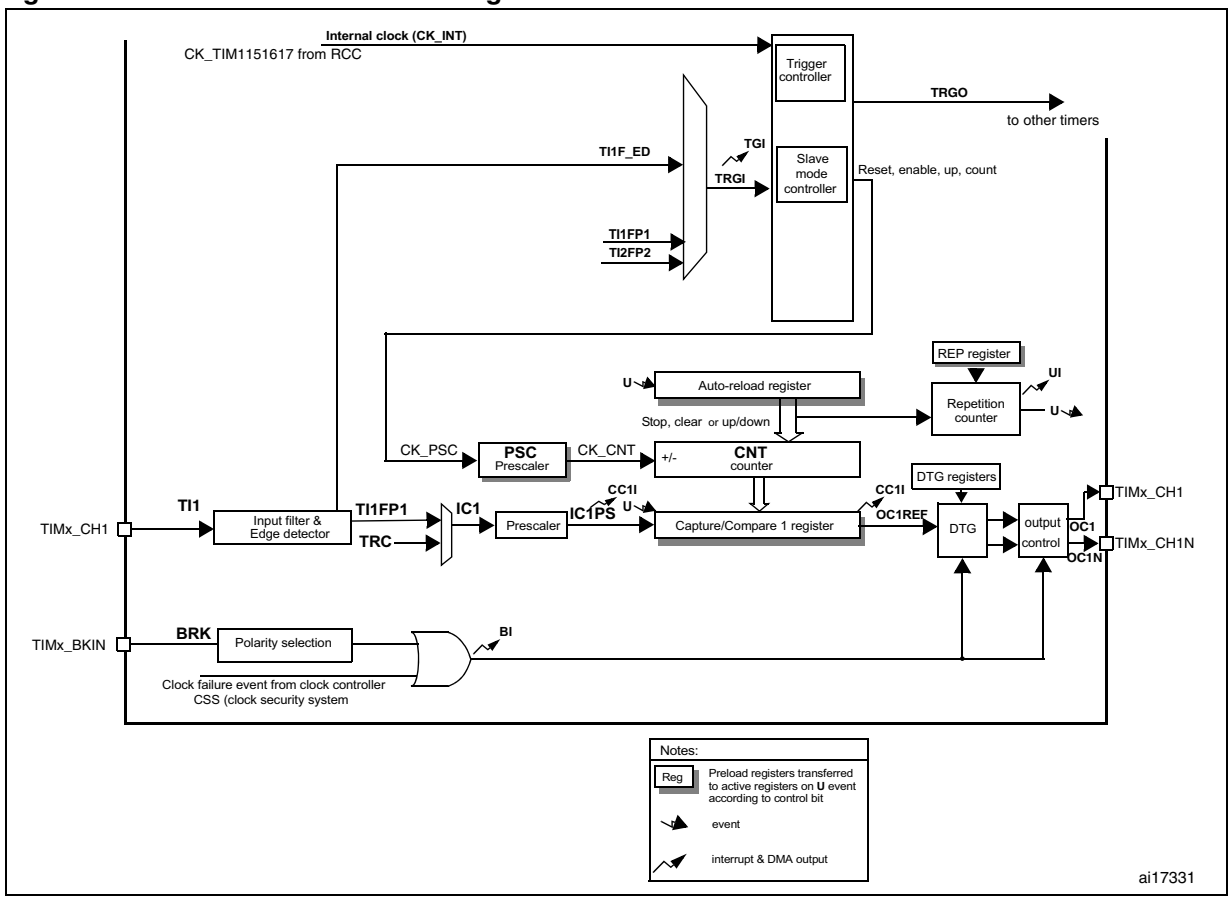


Figure 158. TIM16 and TIM17 block diagram



15.4 TIM15/16/17 functional description

15.4.1 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 137 and *Figure 138* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 159. Counter timing diagram with prescaler division change from 1 to 2

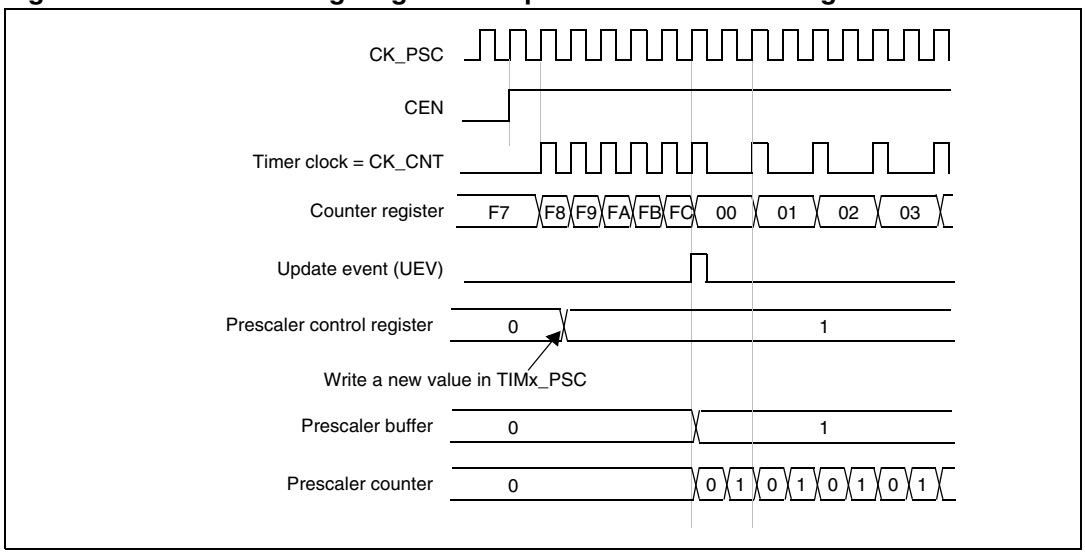
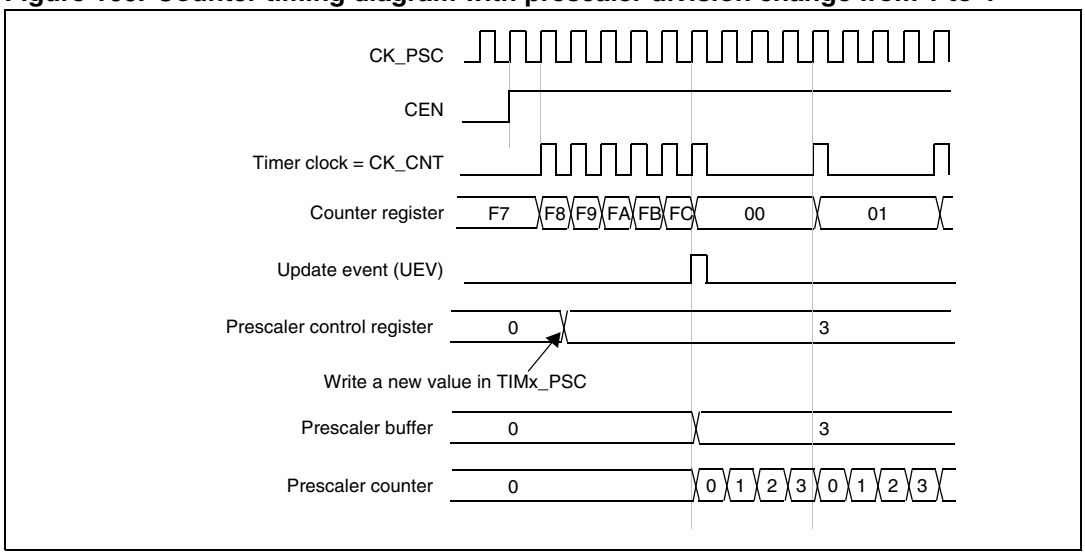


Figure 160. Counter timing diagram with prescaler division change from 1 to 4



15.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the

preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 161. Counter timing diagram, internal clock divided by 1

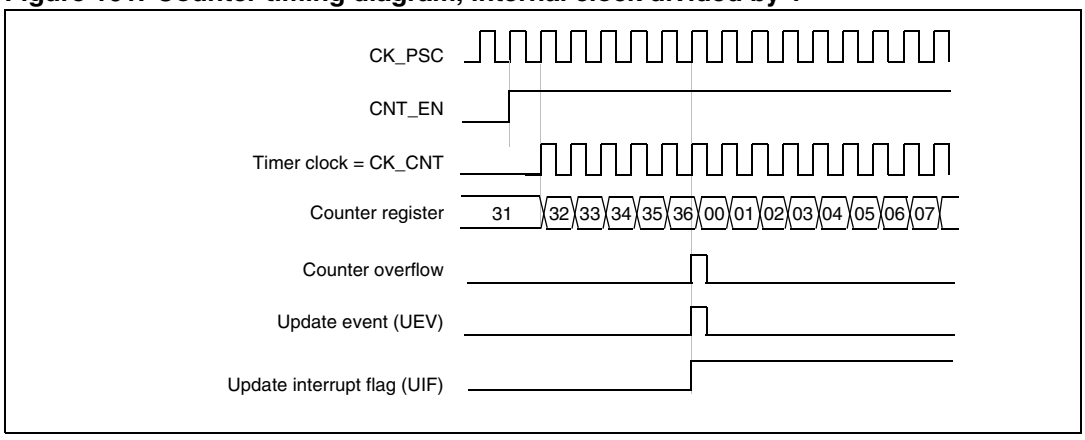


Figure 162. Counter timing diagram, internal clock divided by 2

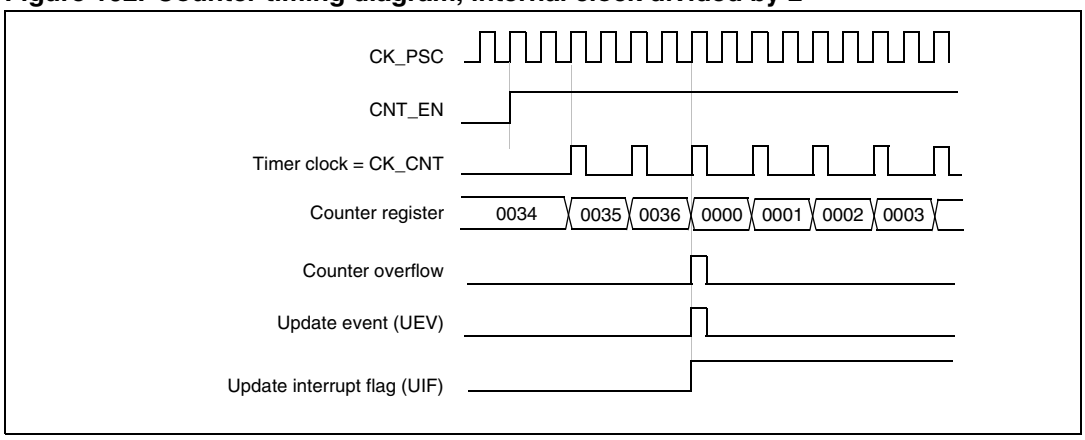


Figure 163. Counter timing diagram, internal clock divided by 4

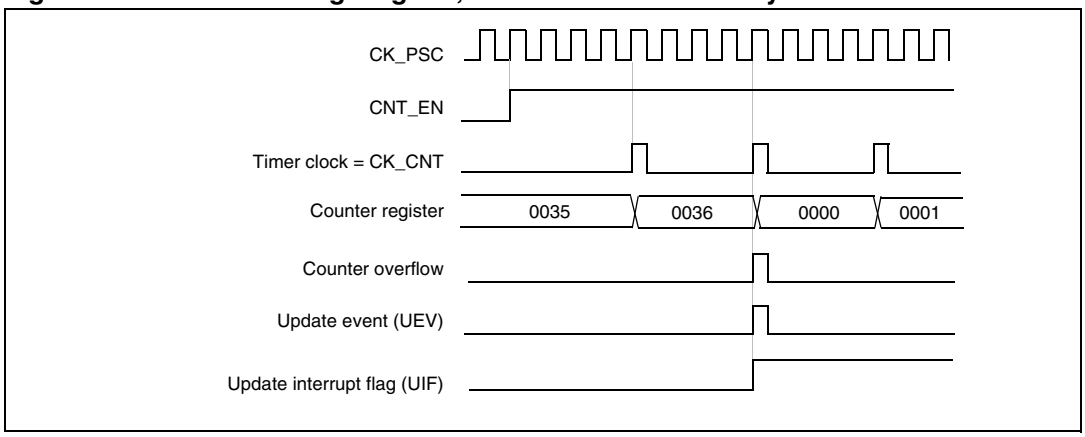


Figure 164. Counter timing diagram, internal clock divided by N

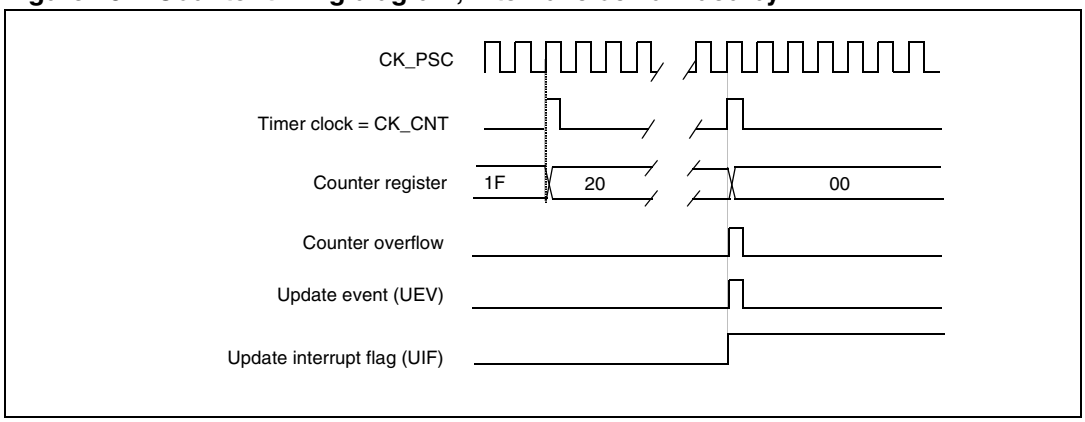


Figure 165. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

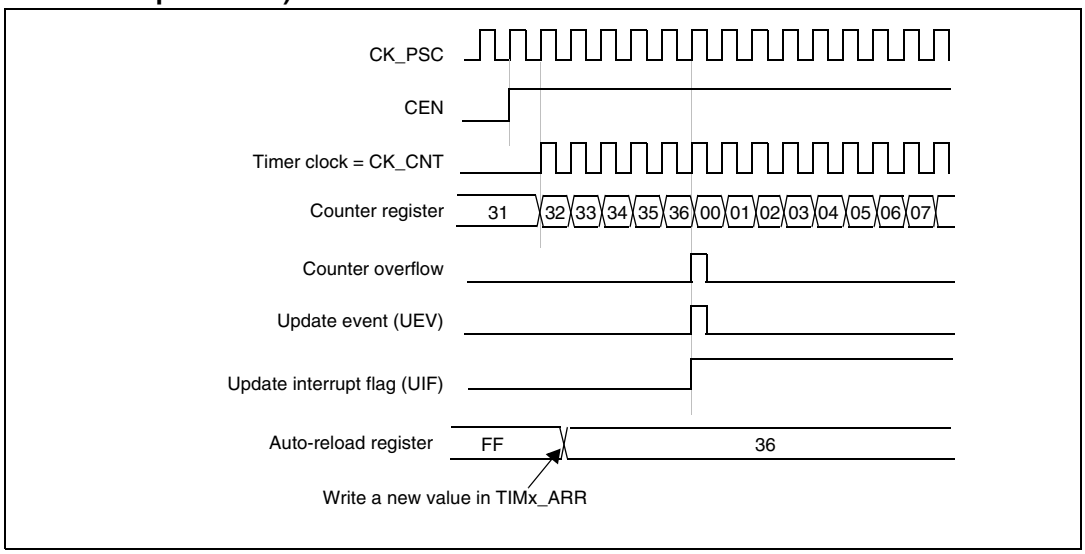
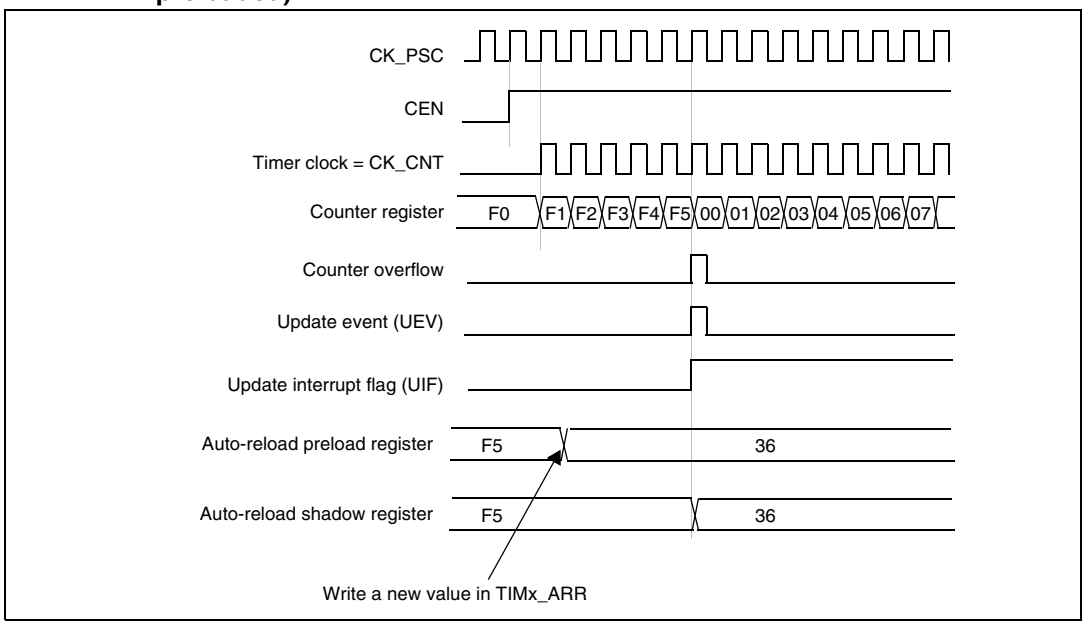


Figure 166. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



15.4.3 Repetition counter

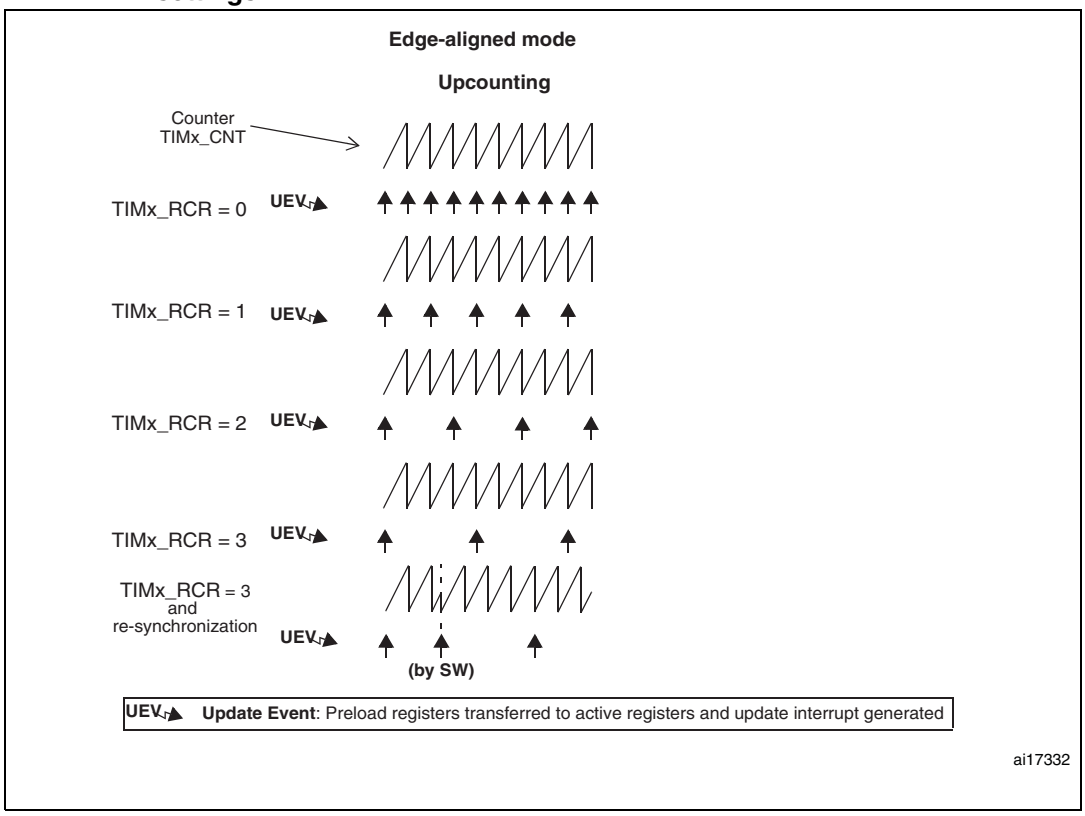
Section 14.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

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 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.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 167*). 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.

Figure 167. Update rate examples depending on mode and TIMx_RCR register settings



15.4.4 Clock selection

The counter clock can be provided by the following clock sources:

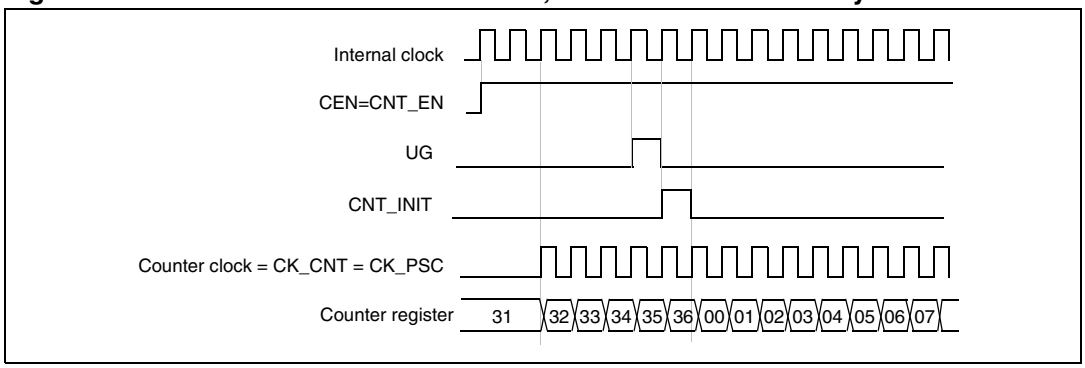
- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, you can configure TIM1 to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another* for more details.

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 144 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

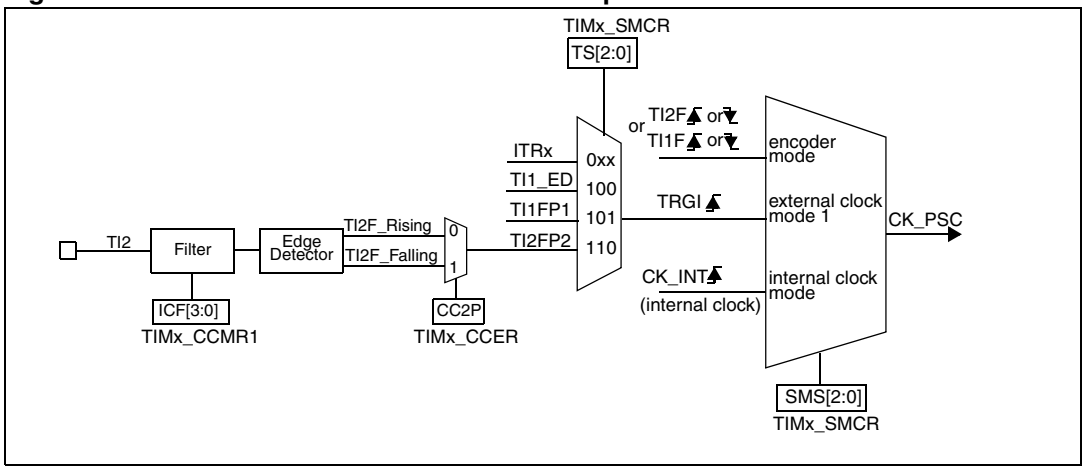
Figure 168. Control circuit in normal mode, internal clock divided by 1



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.

Figure 169. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

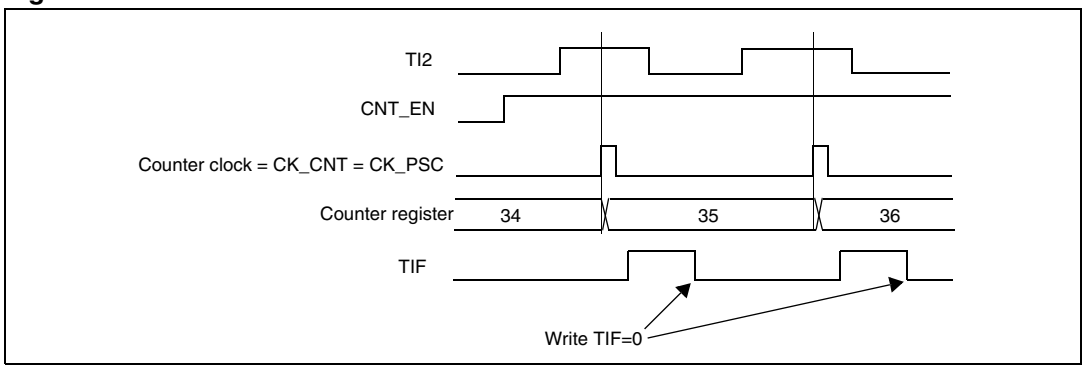
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don’t 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.

Figure 170. Control circuit in external clock mode 1



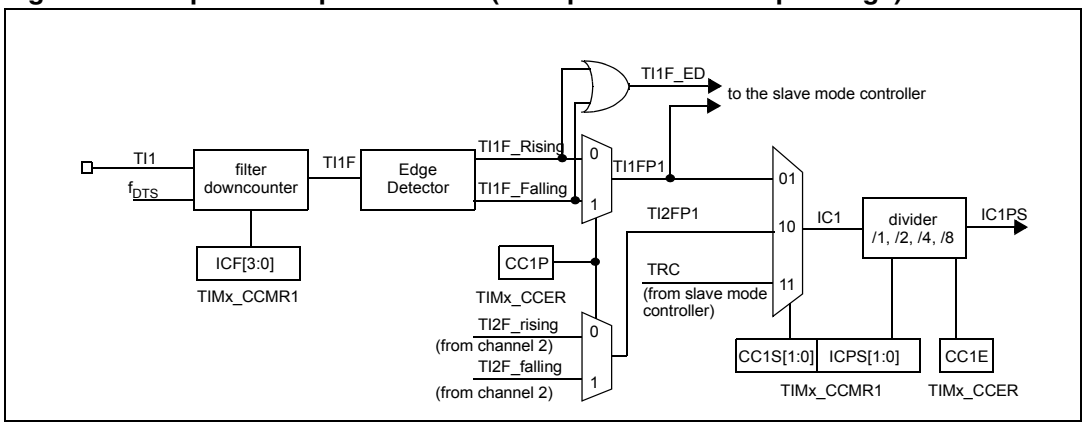
15.4.5 Capture/compare channels

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 147 to *Figure 174* 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 171. 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 172. Capture/compare channel 1 main circuit

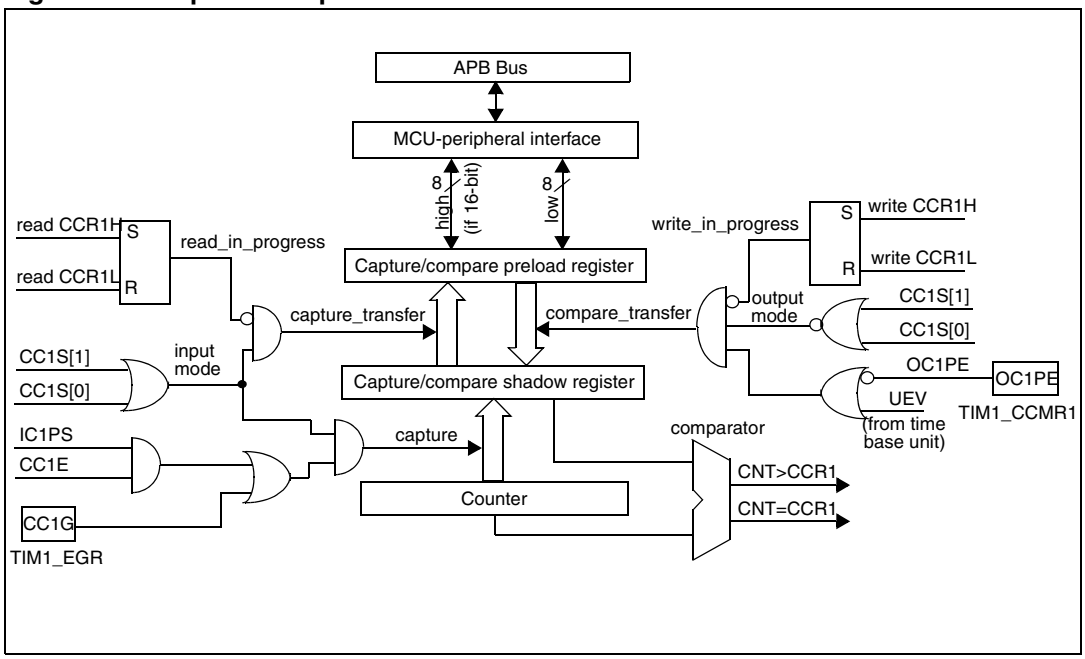


Figure 173. Output stage of capture/compare channel (channel 1)

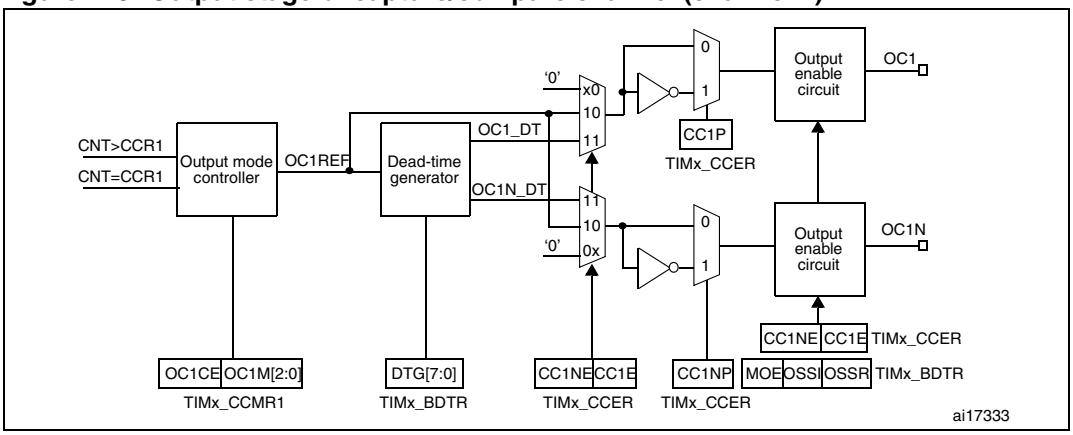
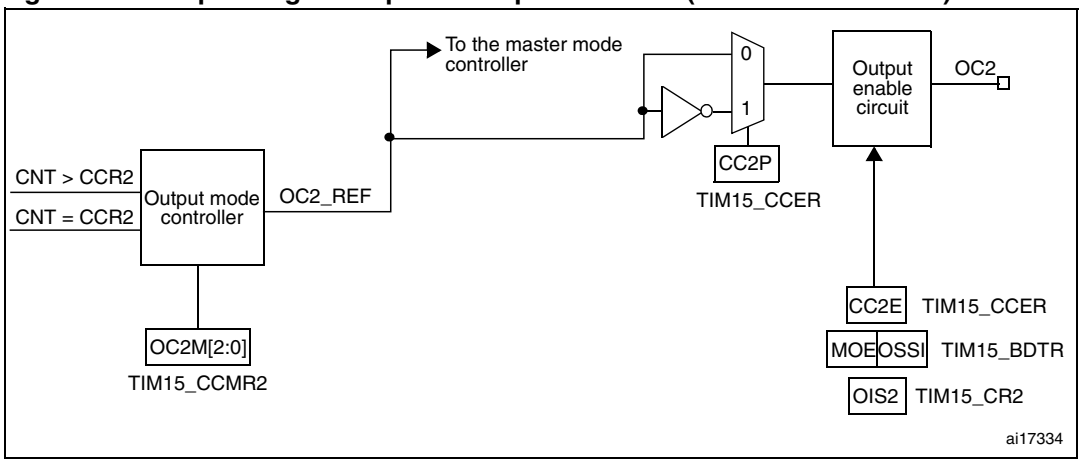


Figure 174. Output stage of capture/compare channel (channel 2 for TIM15)



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.

15.4.6 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to '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:

- 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.
- Program the input filter duration you need with respect to the signal you connect 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 f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- 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).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- 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.

15.4.7 PWM input mode (only for TIM15)

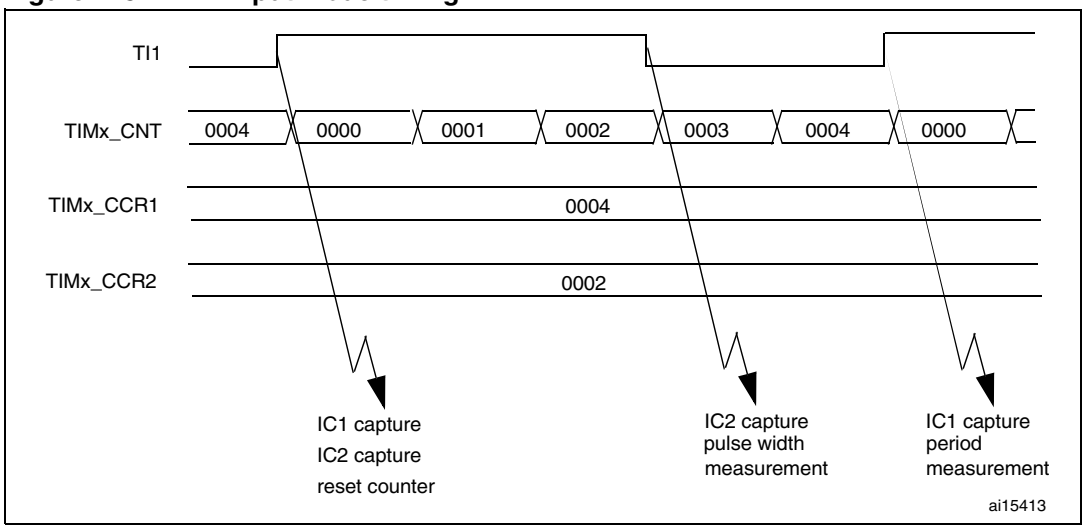
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you 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):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P bit to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx_CCER register.

Figure 175. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

15.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need 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.

15.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

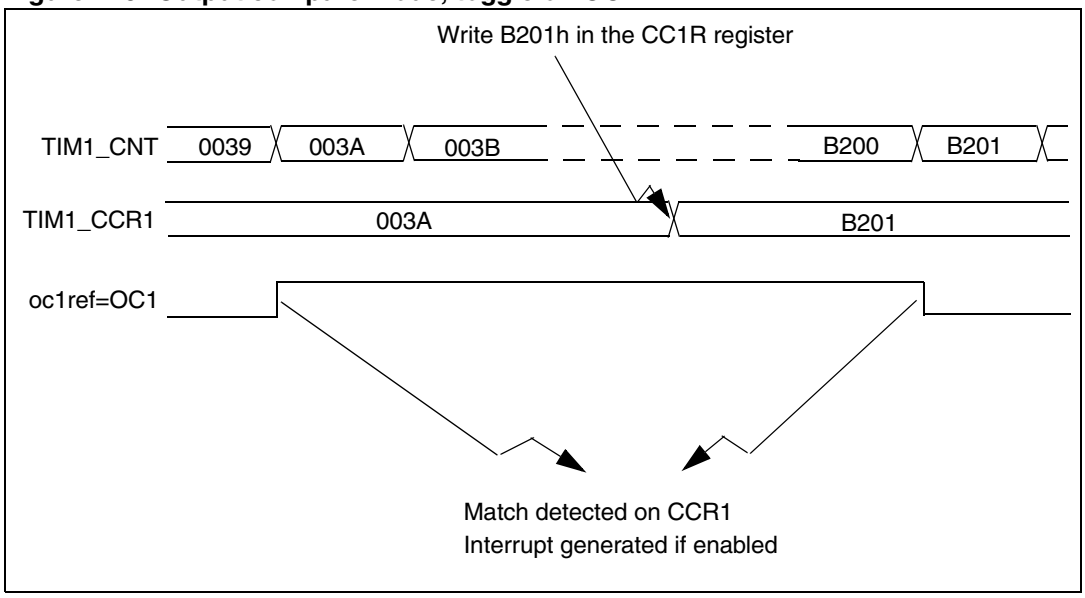
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 151*.

Figure 176. Output compare mode, toggle on OC1.



15.4.10 PWM mode

Pulse Width Modulation mode allows you 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. You must enable the corresponding preload register 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, you have to initialize all the registers 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.

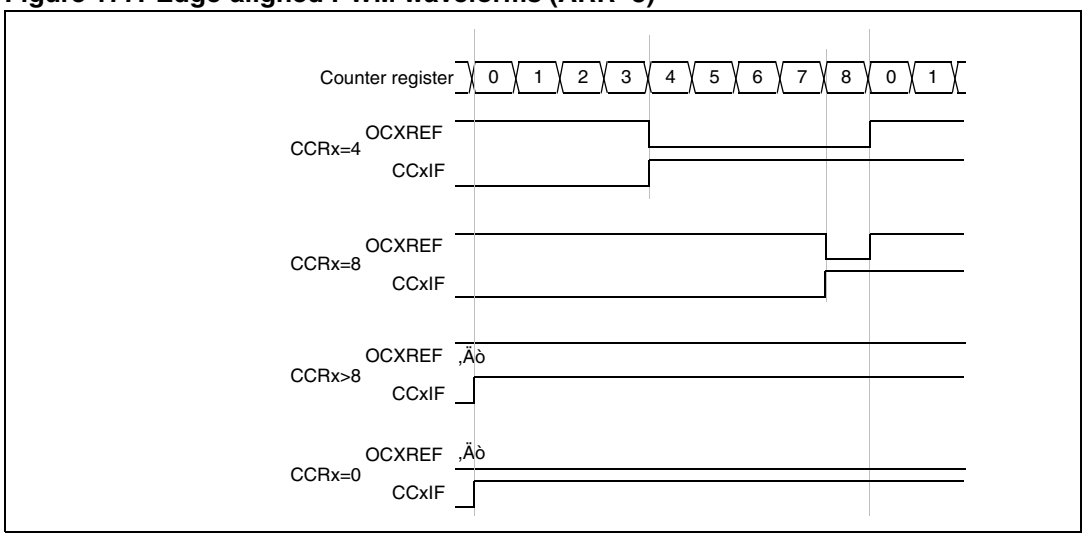
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 337*.

  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 152* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 177. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Repetition counter on page 383*

  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.

15.4.11 Complementary outputs and dead-time insertion

The TIM15/16/17 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 you have to adjust it depending on the devices you have connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

You can select the polarity of the outputs (main output OCx or complementary OCxN) 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 80: Output control bits for complementary OCx and OCxN channels with break feature on page 414* 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 178. Complementary output with dead-time insertion.

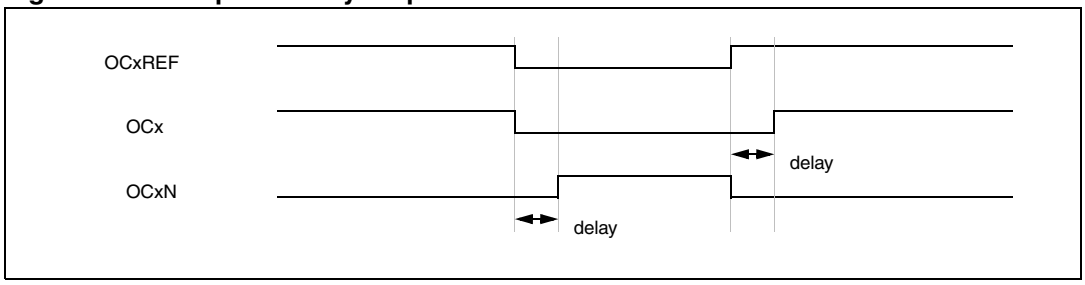


Figure 179. Dead-time waveforms with delay greater than the negative pulse.

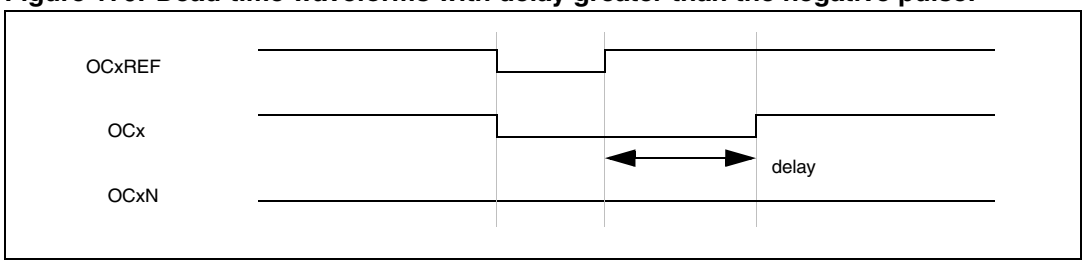
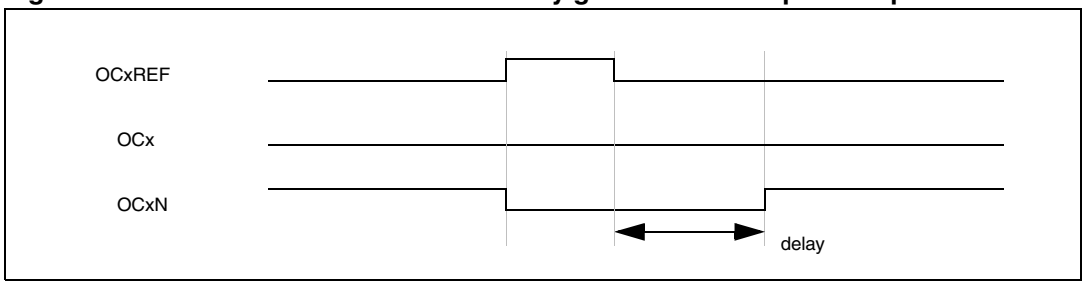


Figure 180. Dead-time waveforms with delay greater than the positive pulse.



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 15.5.15: TIM15 break and dead-time register (TIM15_BDTR) on page 417* 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 you to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

15.4.12 Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the TIMx_BDTR register, OISx and OISxN bits in the TIMx_CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time. Refer to *Table 80: Output control bits for complementary OCx and OCxN channels with break feature on page 414* for more details.

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller. For further information on the Clock Security System, refer to *Section 6.2.7: Clock security system (CSS)*.

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function 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 you write MOE to 1 whereas it was low, you

must insert a delay (dummy instruction) before reading it correctly. This is because you write the asynchronous signal and read 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 in reset state (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 then the timer releases the enable output 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 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. A DMA request can be sent if the BDE 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 you write it to ‘1’ again. In this case, it can be used for security and you can connect the break input to an alarm from power drivers, thermal sensors or any security components.

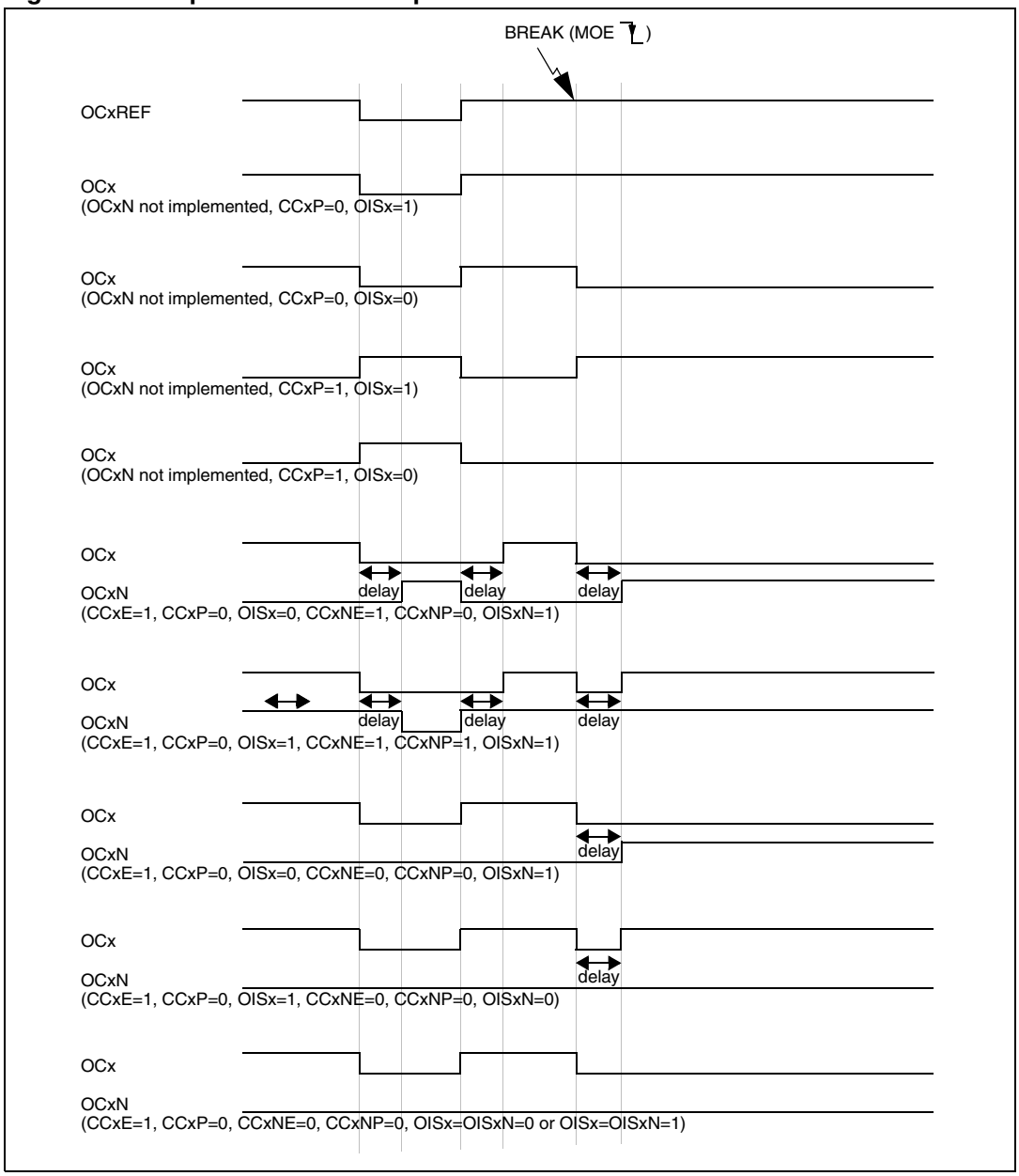
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows you to freeze the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). You can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 15.5.15: TIM15 break and dead-time register (TIM15_BDTR) on page 417*. The LOCK bits can be written only once after an MCU reset.

The *Figure 181* shows an example of behavior of the outputs in response to a break.

Figure 181. Output behavior in response to a break.



15.4.13 One-pulse mode

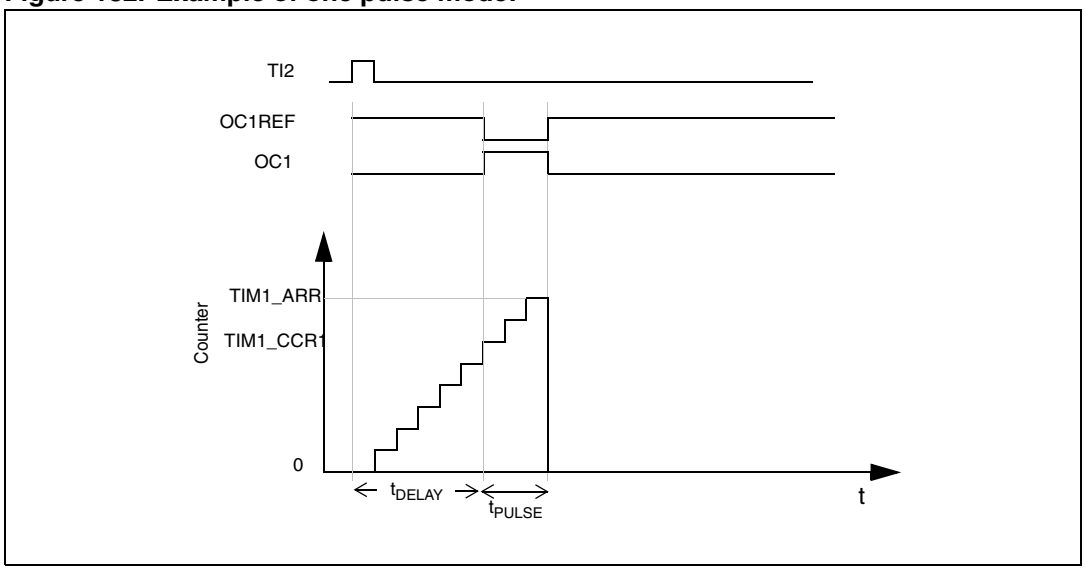
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode 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 ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 182. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the TIMx_SMCR register.
- 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_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say you 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 you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have 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.

You only want 1 pulse, so you write '1' 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_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit 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.

15.4.14 TIM15 and external trigger synchronization (only for TIM15)

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

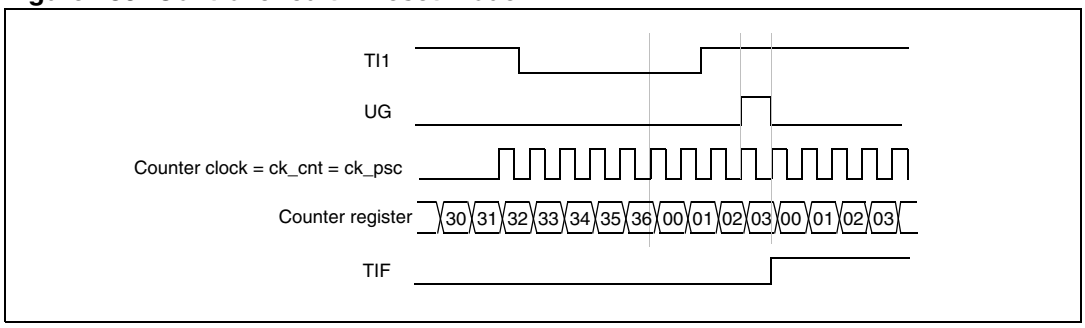
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=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=101 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 183. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

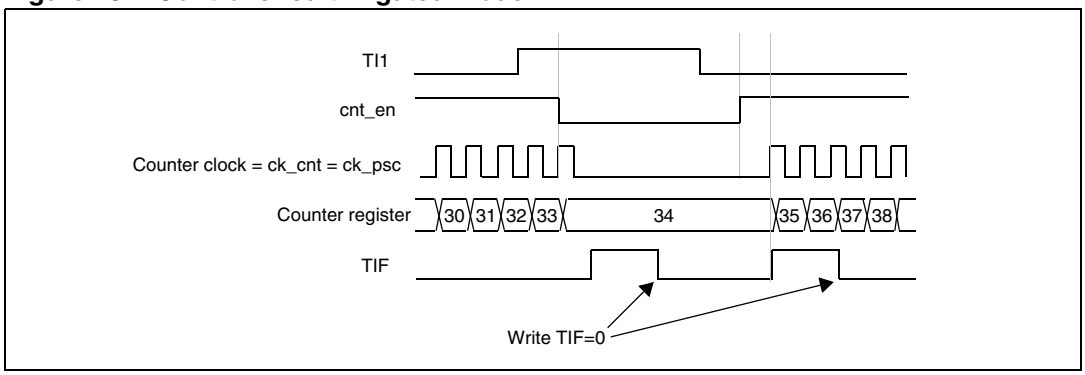
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 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=101 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 184. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

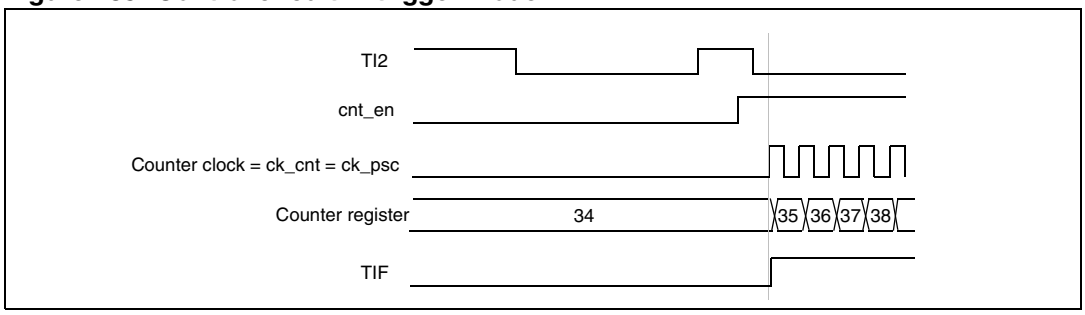
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 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=110 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 185. Control circuit in trigger mode



15.4.15 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 13.3.15: Timer synchronization on page 305* for details.

15.4.16 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 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 25.15.2: Debug support for timers, watchdog and I2C*.

15.5 TIM15 registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

15.5.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

Bits 15:10 Reserved, always read as 0

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (TIx)

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2*t_{CK\_INT}$
10: $t_{DTS} = 4*t_{CK\_INT}$
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, always read as 0

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

15.5.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					OIS2	OIS1N	OIS1	Res.	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw		rw	rw	rw	rw	rw		rw

Bit 15:11 Reserved, always read as 0

Bit 10 **OIS2:** Output idle state 2 (OC2 output)

0: OC2=0 when MOE=0
1: OC2=1 when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIMx_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 7 Reserved, always read as 0

Bits 6:4 **MMS[1:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REF signal is used as trigger output (TRGO).
101: **Compare** - OC2REF signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, always read as 0

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

Note: This bit acts only on channels that have a complementary output.

15.5.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 15:8 Reserved, always read as 0.

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 6:4 **TS[2:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0)
001: Internal Trigger 1 (ITR1)
010: Internal Trigger 2 (ITR2)
011: Internal Trigger 3 (ITR3)
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)

See *Table 79: TIMx Internal trigger connection on page 405* 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 Reserved, always read as 0.

Bits 2:0 **SMS:** 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.
000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.
001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
101: 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.
110: 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.
111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 79. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000)(1)	ITR1 (TS = 001)(1)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM15	TIM2	TIM3	TIM16	TIM17

1. ITR0 and ITR1 triggers available only in high density value line devices.

15.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Reserved			CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Reserved		CC2IE	CC1IE	UIE
	rw				rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, always read as 0.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bits 13:11 Reserved, always read as 0.

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

Bits 4:3 Reserved, always read as 0.

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

15.5.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Reserved		CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0				rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, always read as 0.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag

refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, always read as 0.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 5:3 Reserved, always read as 0.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 15.5.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

15.5.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	Reserved		CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, always read as 0.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, always read as 0.

Bit 2 **CC2G**: Capture/Compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

15.5.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). 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. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		Res	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit 15 Reserved, always read as 0.

Bits 14:12 **OC2M[2:0]**: 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 Reserved, always read as 0.

Bits 6:4 **OC1M**: 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.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.
001: 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).
010: 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).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: 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').
111: 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.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

__2:__ In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).*

__2:__ The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: 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).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: 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 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**: 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**: 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).

15.5.8 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC1NP	Res;	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, always read as 0.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
refer to CC1NP description

Bit 6 Reserved, always read as 0.

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
0: OC1N active high
1: OC1N active low
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).

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
00: noninverted/rising edge: circuit is sensitive to TIxFP1's rising edge (capture, trigger in reset or trigger mode), TIxFP1 is not inverted (trigger in gated mode).
01: inverted/falling edge: circuit is sensitive to TIxFP1's falling edge (capture, trigger in reset, or trigger mode), TIxFP1 is inverted (trigger in gated mode).
10: reserved, do not use this configuration.
11: noninverted/both edges: circuit is sensitive to both the rising and falling edges of TIxFP1 (capture, trigger in reset or trigger mode), TIxFP1 is not inverted (trigger in gated mode).
Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register)..

Bit 0 **CC1E**: Capture/Compare 1 output enable
CC1 channel configured as output:
0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 80. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
		1	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
0	0	X	0	0	Output Disabled (not driven by the timer) Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state.	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
	1		1	0		
	1		1	1		

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), 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 and AFIO registers.

15.5.9 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

15.5.10 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

15.5.11 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 14.4.1: Time-base unit on page 336* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

15.5.12 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, always read as 0.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

15.5.13 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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).

15.5.14 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 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 signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

15.5.15 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

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

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 15.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 412*).

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).

Note: 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).

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 15.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 412*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1. Then, 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).

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 15.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 412*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

15.5.16 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
Res.			rw	rw	rw	rw	rw	Res.			rw	rw	rw	rw	rw

Bits 15:13 Reserved, always read as 0

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).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, always read as 0

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

15.5.17 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
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).

15.5.18 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 81. TIM15 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM15_CR1	Reserved																						CKD [1:0]		ARPE	Reserved				OPM	URS	UDIS	CEN
	Reset value																							0	0	0					0	0	0	0
0x04	TIM15_CR2	Reserved																					OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0
0x08	TIM15_SMCR	Reserved																								MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																									0	0	0	0		0	0	0
0x0C	TIM15_DIER	Reserved																	TDE	Reserved			CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Reserved		CC2IE	CC1IE	UIE
	Reset value																		0				0	0	0	0	0	0			0	0	0
0x10	TIM15_SR	Reserved																					CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	Reserved		CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	TIM15_EGR	Reserved																								BG	TG	COMG	Reserved		CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	TIM15_CCMR1 Output Compare mode	Reserved																	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Reserved	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIM15_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIM15_CCER	Reserved																								CC2NP	Reserved	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	TIM15_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIM15_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 81. TIM15 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
0x2C	TIM15_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TIM15_RCR	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIM15_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIM15_CCR2	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIM15_BDTR	Reserved																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIM15_DCR	Reserved																			DBL[4:0]					Reserved		DBA[4:0]				
	Reset value																				0	0	0	0	0			0	0	0	0	0
0x4C	TIM15_DMAR	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

15.6 TIM16&TIM17 registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

15.6.1 TIM16&TIM17 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

Bits 15:10 Reserved, always read as 0

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 (t_{DTS})used by the dead-time generators and the digital filters (TIx),

00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
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, always read as 0.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

15.6.2 TIM16&TIM17 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						OIS1N	OIS1	Reserved				CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, always read as 0

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, always read as 0

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, always read as 0

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

Note: This bit acts only on channels that have a complementary output.

15.6.3 TIM16&TIM17 DMA/interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Reserved				CC1DE	UDE	BIE	TIE	COMIE	Reserved			CC1IE	UIE
	rw					rw	rw	rw	rw	rw				rw	rw

Bit 15 Reserved, always read as 0.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bist 13:10 Reserved, always read as 0.

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

Bits 4:2 Reserved, always read as 0.

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

15.6.4 TIM16&TIM17 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Res.	BIF	TIF	COMIF	Reserved			CC1IF	UIF
						rc_w0		rc_w0	rc_w0	rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, always read as 0.

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, always read as 0.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, always read as 0.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

If channel CC1 is configured as output:
This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.
0: No match.
1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.
0: No input capture occurred
1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by a trigger event (refer to *Section 15.5.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

15.6.5 TIM16&TIM17 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	Reserved			CC1G	UG
								w	w	w				w	w

Bits 15:8 Reserved, always read as 0.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, always read as 0.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

15.6.6 TIM16&TIM17 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). 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. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Res	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 15:7 Reserved, always read as 0.

Bits 6:4 **OC1M**: 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.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.
001: 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).
010: 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).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: 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’).
111: 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.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).*

***2:** In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

*Note: **1:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).*

***2:** The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: 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).

Input capture mode

Bits 15:8 Reserved, always read as 0.

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 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=
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**: 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**: 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).

15.6.7 TIM16&TIM17 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, always read as 0.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

0: OC1N active high
1: OC1N active low

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).

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 for capture operation.

00: Non-inverted/rising edge: circuit is sensitive to TIxFP1's rising edge TIxFP1 is not inverted.

01: Inverted/falling edge: circuit is sensitive to TIxFP1's falling edge, TIxFP1 is inverted.

10: Reserved, do not use this configuration.

11: Non-inverted/both edges: circuit is sensitive to both the rising and falling edges of TIxFP1, TIxFP1 is not inverted.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register)

Bit 0 **CC1E**: Capture/Compare 1 output enable

CC1 channel configured as output:
0: Off - OC1 is not active. OC1 level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.
1: On - OC1 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits.

CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 82. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer) OCx=0, OCx_EN=0	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
		1	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
0	0	X	0	0	Output Disabled (not driven by the timer) Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state.	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
	1		1	0		
	1		1	1		

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), 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 and AFIO registers.

15.6.8 TIM16&TIM17 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

15.6.9 TIM16&TIM17 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

15.6.10 TIM16&TIM17 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 14.4.1: Time-base unit on page 336* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

15.6.11 TIM16&TIM17 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, always read as 0.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

15.6.12 TIM16&TIM17 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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).

15.6.13 TIM16&TIM17 break and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

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
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 15.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 412*).

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).

Note: 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).

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 15.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 412*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1. Then, 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).

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 15.5.8: TIM15 capture/compare enable register (TIM15_CCER) on page 412*).
0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)
1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.
00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

15.6.14 TIM16&TIM17 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, always read as 0

Bits 12:8 **DBL[4:0]**: DMA burst length
This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, always read as 0

Bits 4:0 **DBA[4:0]**: DMA base address
This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

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..

15.6.15 TIM16&TIM17 DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
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 access to the DMAR register accesses the register located at the address:
"(TIMx_CR1 address) + DBA + (DMA index)" in which:
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 the offset automatically controlled by the DMA transfer, depending on the length of the transfer DBL in the TIMx_DCR register.

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) 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

Note: 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.

15.6.16 TIM16&TIM17 register map

TIM16&TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 83. TIM16&TIM17 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																							0	0	0				0	0	0	0
0x04	TIMx_CR2	Reserved																						OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																							0	0	0	0	0	0	0	0		0
0x0C	TIMx_DIER	Reserved																	TDE	Reserved				CC1DE	UDE	BIE	TIE	Reserved				CC1IE	UIE
	Reset value																		0					0	0	0	0					0	0
0x10	TIMx_SR	Reserved																						CC1OF	Reserved	BIF	TIF	Reserved				CC1IF	UIF
	Reset value																							0		0	0					0	0
0x14	TIMx_EGR	Reserved																								BG	TG	COMG	Reserved			CC1G	UG
	Reset value																									0	0	0				0	0
0x18	TIMx_CCMR1 Output Compare mode	Reserved																									OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved																								IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	TIMx_CCER	Reserved																												CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TIMx_RCR	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Reserved																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Reserved																			DBL[4:0]					Reserved	DBA[4:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0

Table 83. TIM16&TIM17 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
0x4C	TIMx_DMAR	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

16 Basic timers (TIM6&TIM7)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

16.1 TIM6&TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

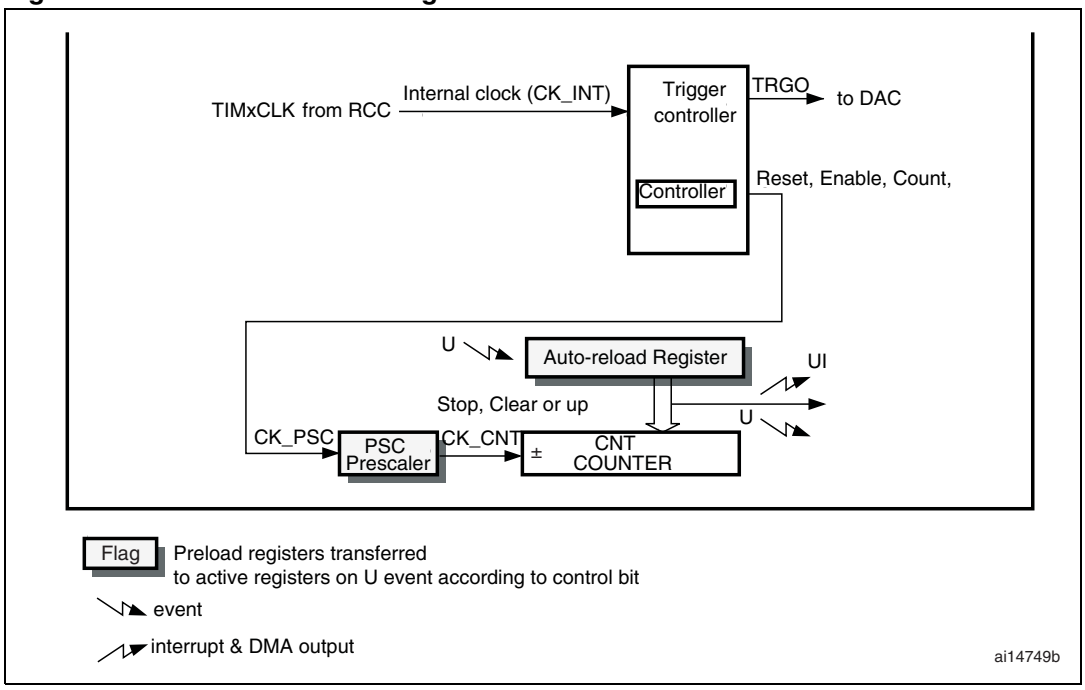
The timers are completely independent, and do not share any resources.

16.2 TIM6&TIM7 main features

Basic timer (TIM6&TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 186. Basic timer block diagram



16.3 TIM6&TIM7 functional description

16.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 187 and *Figure 188* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 187. Counter timing diagram with prescaler division change from 1 to 2

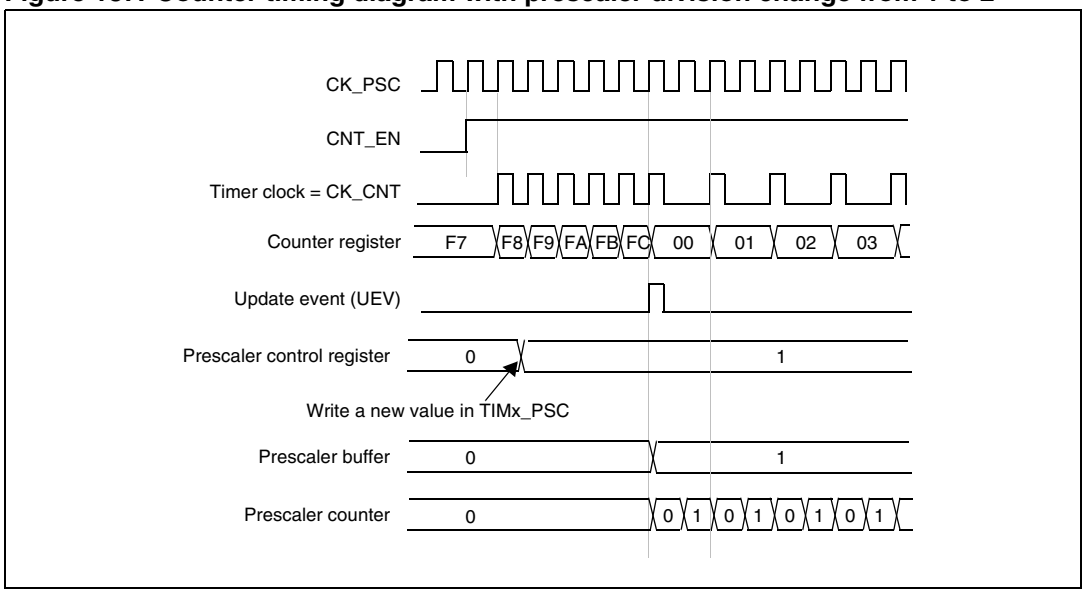
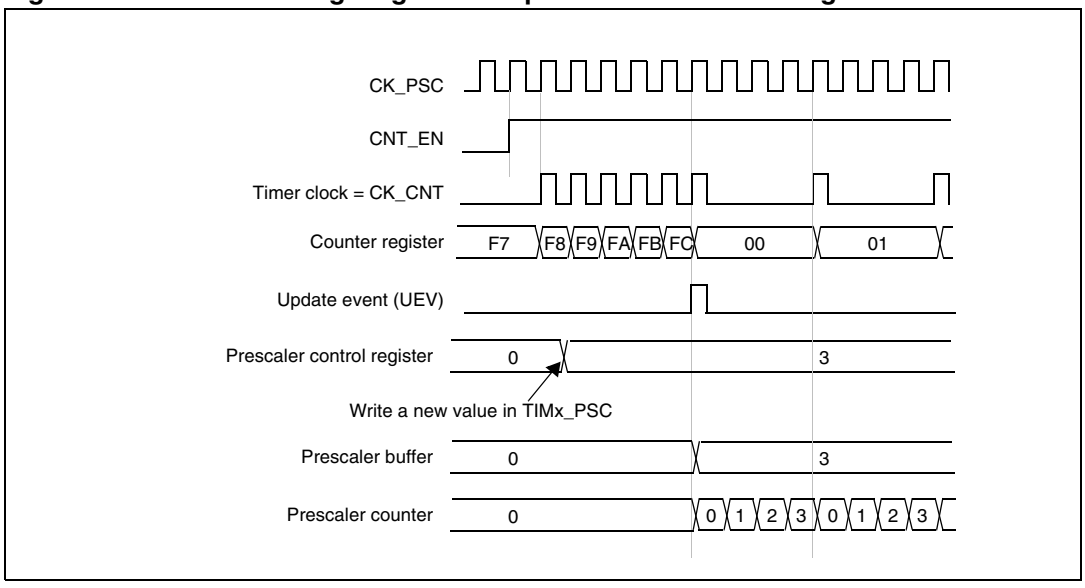


Figure 188. Counter timing diagram with prescaler division change from 1 to 4



16.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 189. Counter timing diagram, internal clock divided by 1

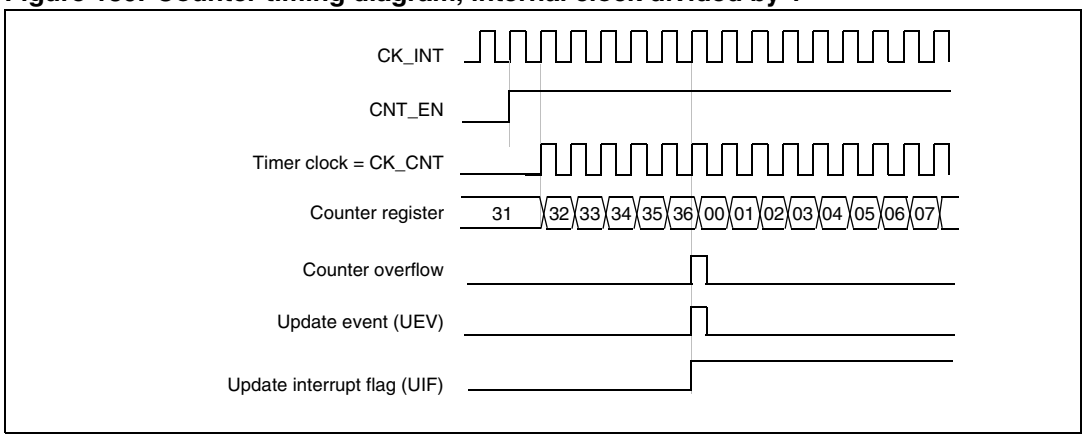


Figure 190. Counter timing diagram, internal clock divided by 2

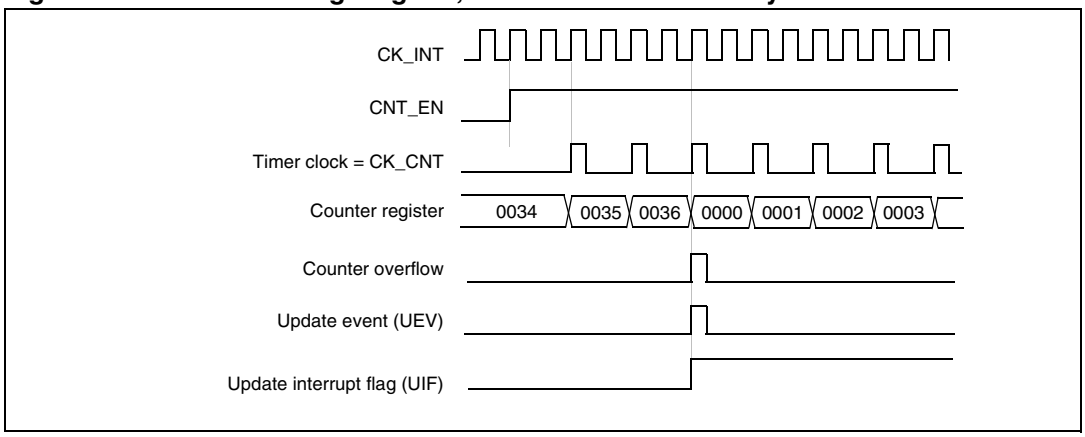


Figure 191. Counter timing diagram, internal clock divided by 4

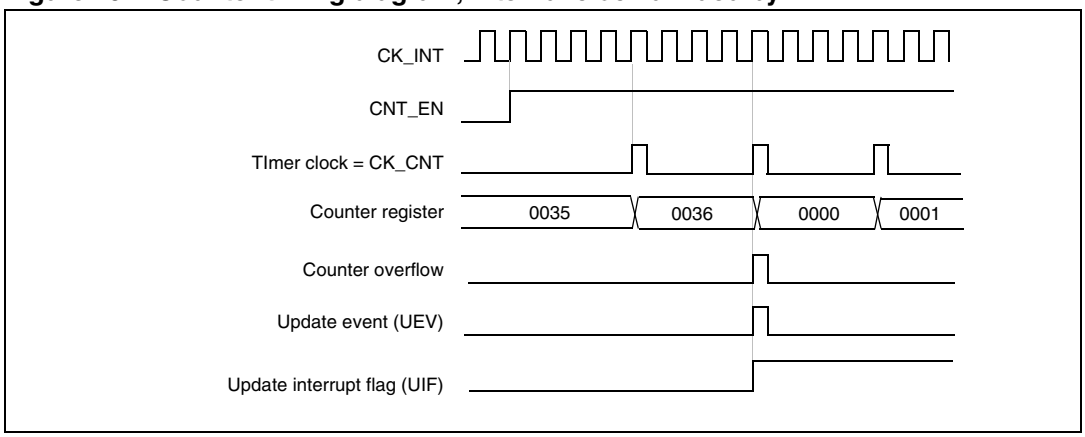


Figure 192. Counter timing diagram, internal clock divided by N

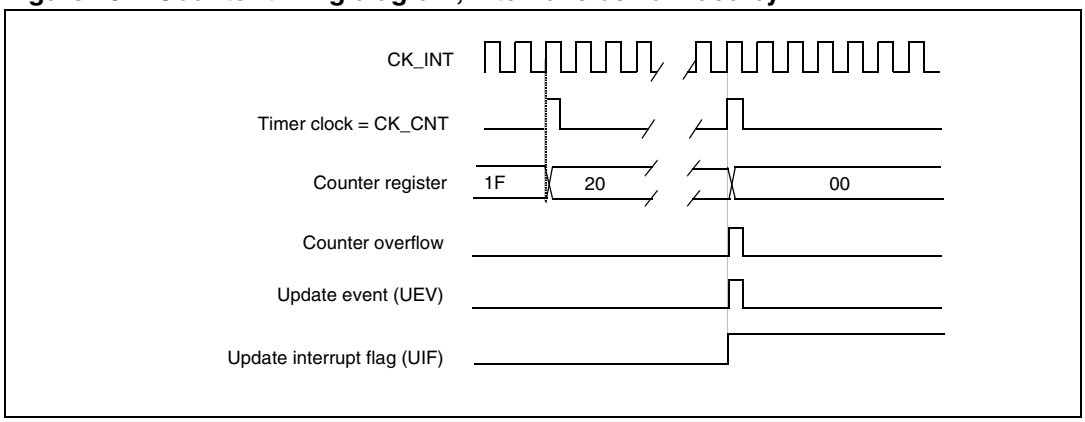


Figure 193. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

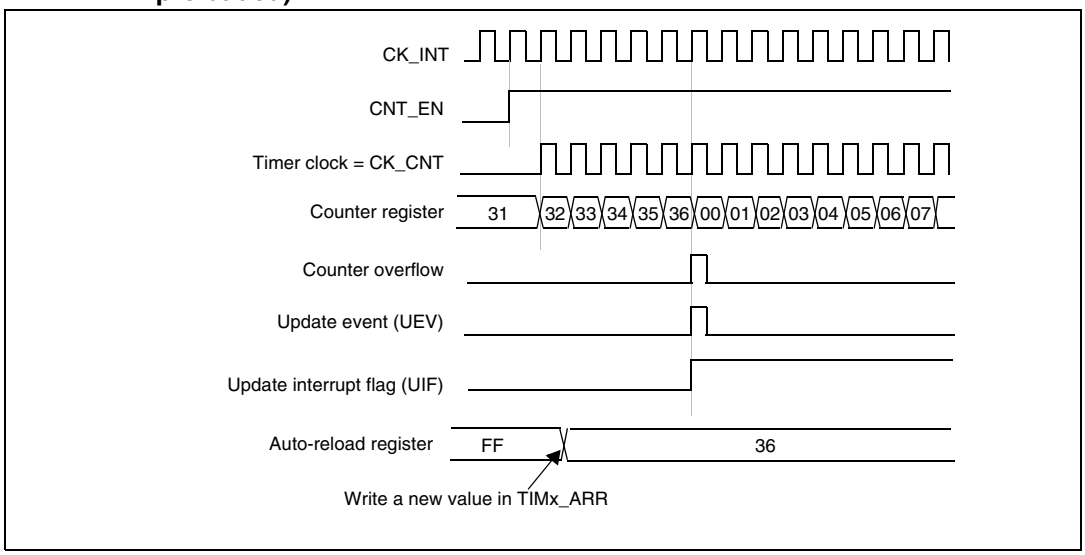
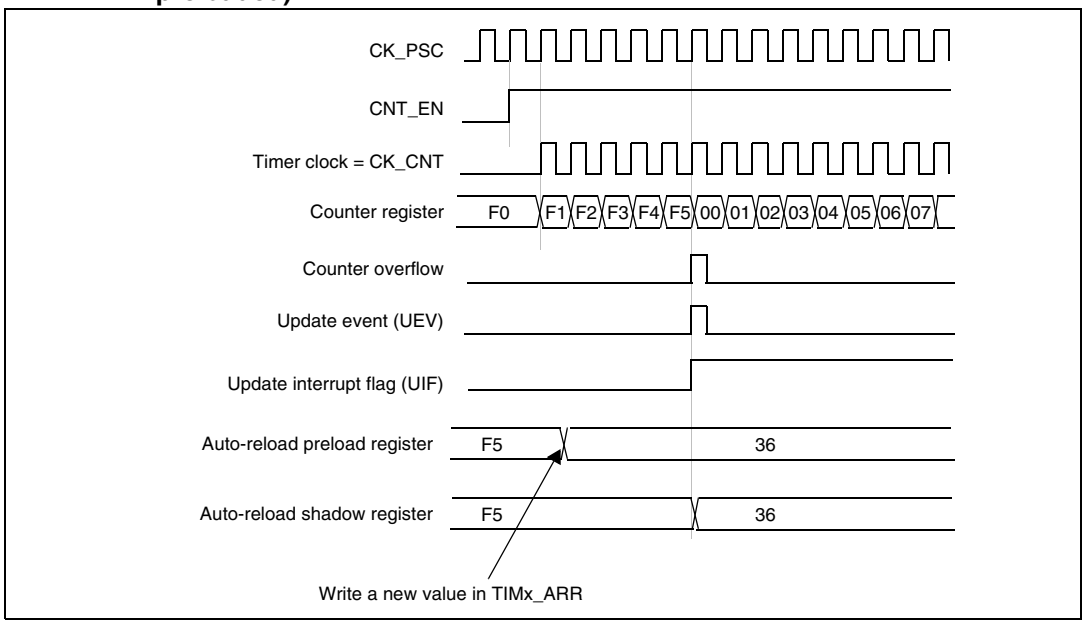


Figure 194. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



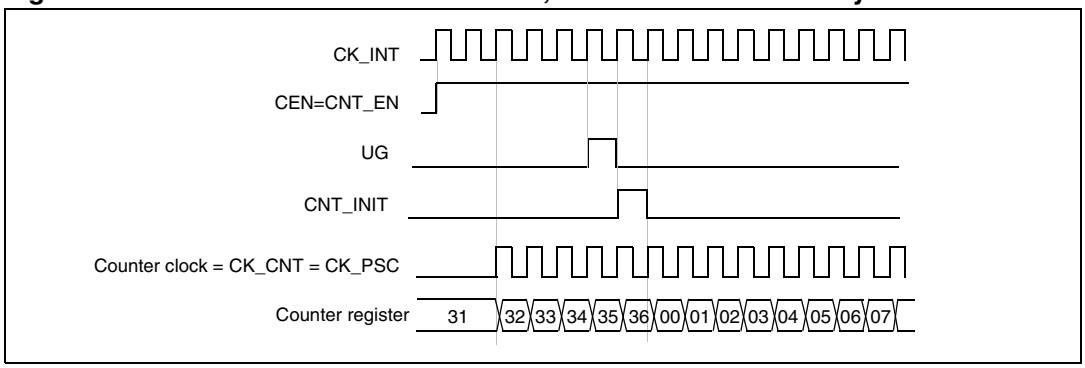
16.3.3 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 195 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 195. Control circuit in normal mode, internal clock divided by 1



16.3.4 Debug mode

When the microcontroller enters the debug mode (Cortex™-M3 core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBG module. For more details, refer to *Section 25.15.2: Debug support for timers, watchdog and I2C*.

16.4 TIM6&TIM7 registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

16.4.1 TIM6&TIM7 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

Bits 15:8 Reserved, always read as 0

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, always read as 0

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

16.4.2 TIM6&TIM7 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									MMS[2:0]			Reserved			
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (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 a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

Bits 3:0 Reserved, always read as 0

16.4.3 TIM6&TIM7 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							UDE	Reserved							UIE
							rw								rw

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

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

16.4.4 TIM6&TIM7 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

16.4.5 TIM6&TIM7 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

16.4.6 TIM6&TIM7 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CNT[15:0]**: Counter value

16.4.7 TIM6&TIM7 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event.

16.4.8 TIM6&TIM7 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 16.3.1: Time-base unit on page 441* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

16.4.9 TIM6&TIM7 register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 84. TIM6&TIM7 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	**TIMx_CR2**	Reserved																									MMS[2:0]			Reserved				
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	**TIMx_DIER**	Reserved																							UDE	Reserved							UIE
	Reset value																								0								0
0x10	**TIMx_SR**	Reserved																															UIF
	Reset value																																0
0x14	**TIMx_EGR**	Reserved																															UG
	Reset value																																0
0x18	Reserved																																
0x1C	Reserved																																
0x20	Reserved																																
0x24	**TIMx_CNT**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 1 on page 36* and *Table 2 on page 37* for the register boundary addresses.

17 Real-time clock (RTC)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F10xxx family, unless otherwise specified.

17.1 RTC introduction

The real-time clock is an independent timer. The RTC provides a set of continuously running counters which can be used, with suitable software, to provide a clock-calendar function. The counter values can be written to set the current time/date of the system.

The RTC core and clock configuration (RCC_BDCR register) are in the Backup domain, which means that RTC setting and time are kept after reset or wakeup from Standby mode.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:

- enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register
- set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

17.2 RTC main features

- Programmable prescaler: division factor up to 2^{20}
- 32-bit programmable counter for long-term measurement
- Two separate clocks: PCLK1 for the APB1 interface and RTC clock (must be at least four times slower than the PCLK1 clock)
- The RTC clock source could be any of the following ones:
  - HSE clock divided by 128
  - LSE oscillator clock
  - LSI oscillator clock (refer to *Section 6.2.8: RTC clock* for details)
- Two separate reset types:
  - The APB1 interface is reset by system reset
  - The RTC Core (Prescaler, Alarm, Counter and Divider) is reset only by a Backup domain reset (see *Section 6.1.3: Backup domain reset on page 69*).
- Three dedicated maskable interrupt lines:
  - Alarm interrupt, for generating a software programmable alarm interrupt.
  - Seconds interrupt, for generating a periodic interrupt signal with a programmable period length (up to 1 second).
  - Overflow interrupt, to detect when the internal programmable counter rolls over to zero.

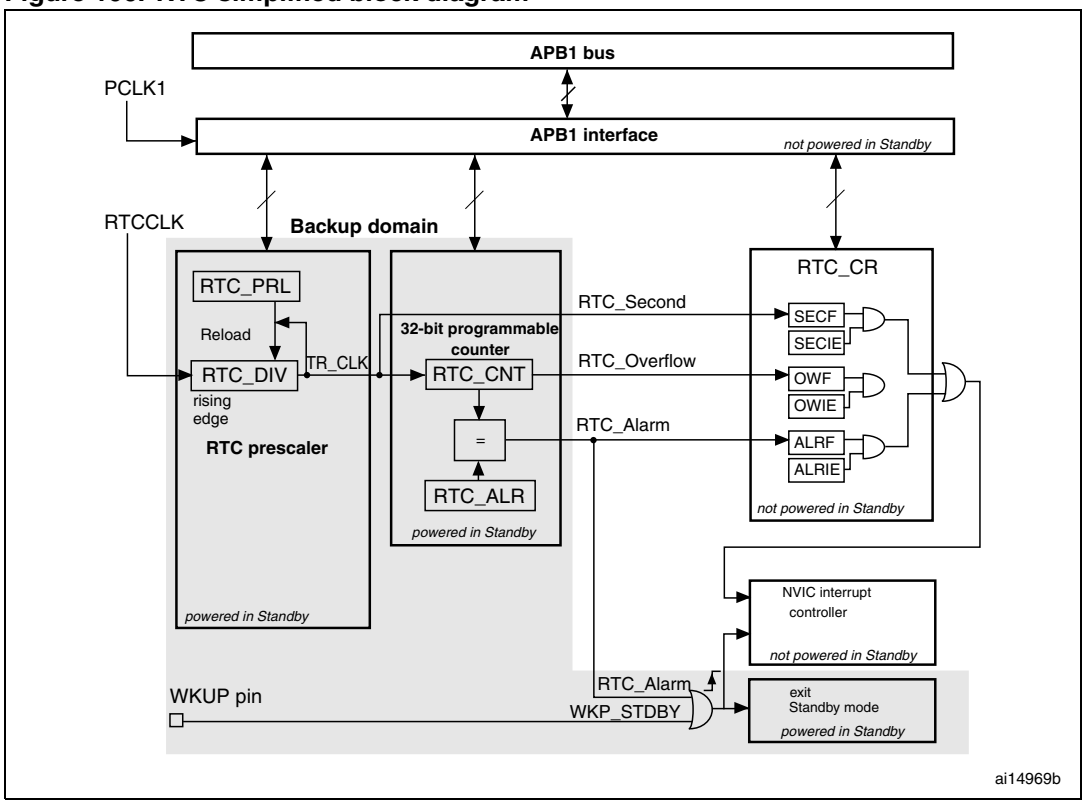
17.3 RTC functional description

17.3.1 Overview

The RTC consists of two main units (see *Figure 196 on page 454*). The first one (APB1 Interface) is used to interface with the APB1 bus. This unit also contains a set of 16-bit registers accessible from the APB1 bus in read or write mode (for more information refer to *Section 17.4: RTC registers on page 457*). The APB1 interface is clocked by the APB1 bus clock in order to interface with the APB1 bus.

The other unit (RTC Core) consists of a chain of programmable counters made of two main blocks. The first block is the RTC prescaler block, which generates the RTC time base TR_CLK that can be programmed to have a period of up to 1 second. It includes a 20-bit programmable divider (RTC Prescaler). Every TR_CLK period, the RTC generates an interrupt (Second Interrupt) if it is enabled in the RTC_CR register. The second block is a 32-bit programmable counter that can be initialized to the current system time. The system time is incremented at the TR_CLK rate and compared with a programmable date (stored in the RTC_ALR register) in order to generate an alarm interrupt, if enabled in the RTC_CR control register.

Figure 196. RTC simplified block diagram



17.3.2 Resetting RTC registers

All system registers are asynchronously reset by a System Reset or Power Reset, except for RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV.

The RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV registers are reset only by a Backup Domain reset. Refer to *Section 6.1.3: Backup domain reset*.

17.3.3 Reading RTC registers

The RTC core is completely independent from the RTC APB1 interface.

Software accesses the RTC prescaler, counter and alarm values through the APB1 interface but the associated readable registers are internally updated at each rising edge of the RTC clock resynchronized by the RTC APB1 clock. This is also true for the RTC flags.

This means that the first read to the RTC APB1 registers may be corrupted (generally read as 0) if the APB1 interface has previously been disabled and the read occurs immediately after the APB1 interface is enabled but before the first internal update of the registers. This can occur if:

- A system reset or power reset has occurred
- The MCU has just woken up from Standby mode (see *Section 4.3: Low-power modes*)
- The MCU has just woken up from Stop mode (see *Section 4.3: Low-power modes*)

In all the above cases, the RTC core has been kept running while the APB1 interface was disabled (reset, not clocked or unpowered).

Consequently when reading the RTC registers, after having disabled the RTC APB1 interface, the software must first wait for the RSF bit (Register Synchronized Flag) in the RTC_CRL register to be set by hardware.

Note that the RTC APB1 interface is not affected by WFI and WFE low-power modes.

17.3.4 Configuring RTC registers

To write in the RTC_PRL, RTC_CNT, RTC_ALR registers, the peripheral must enter Configuration Mode. This is done by setting the CNF bit in the RTC_CRL register.

In addition, writing to any RTC register is only enabled if the previous write operation is finished. To enable the software to detect this situation, the RTOFF status bit is provided in the RTC_CR register to indicate that an update of the registers is in progress. A new value can be written to the RTC registers only when the RTOFF status bit value is ’1’.

Configuration procedure:

1. Poll RTOFF, wait until its value goes to ‘1’
2. Set the CNF bit to enter configuration mode
3. Write to one or more RTC registers
4. Clear the CNF bit to exit configuration mode
5. Poll RTOFF, wait until its value goes to ‘1’ to check the end of the write operation.

The write operation only executes when the CNF bit is cleared; it takes at least three RTCCLK cycles to complete.

17.3.5 RTC flag assertion

The RTC Second flag (SECF) is asserted on each RTC Core clock cycle before the update of the RTC Counter.

The RTC Overflow flag (OWF) is asserted on the last RTC Core clock cycle before the counter reaches 0x0000.

The RTC_Alarm and RTC Alarm flag (ALRF) (see *Figure 197*) are asserted on the last RTC Core clock cycle before the counter reaches the RTC Alarm value stored in the Alarm register increased by one (RTC_ALR + 1). The write operation in the RTC Alarm and RTC Second flag must be synchronized by using one of the following sequences:

- Use the RTC Alarm interrupt and inside the RTC interrupt routine, the RTC Alarm and/or RTC Counter registers are updated.
- Wait for SECF bit to be set in the RTC Control register. Update the RTC Alarm and/or the RTC Counter register.

Figure 197. RTC second and alarm waveform example with PR=0003, ALARM=00004

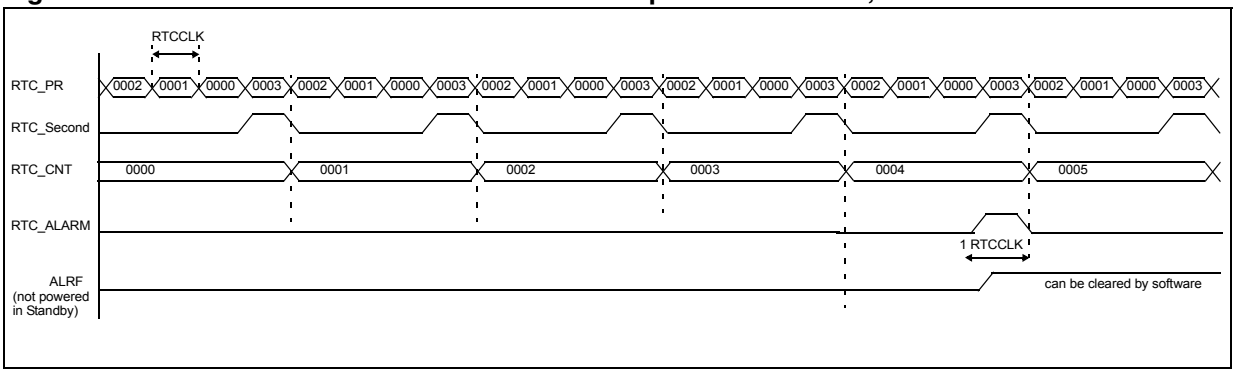
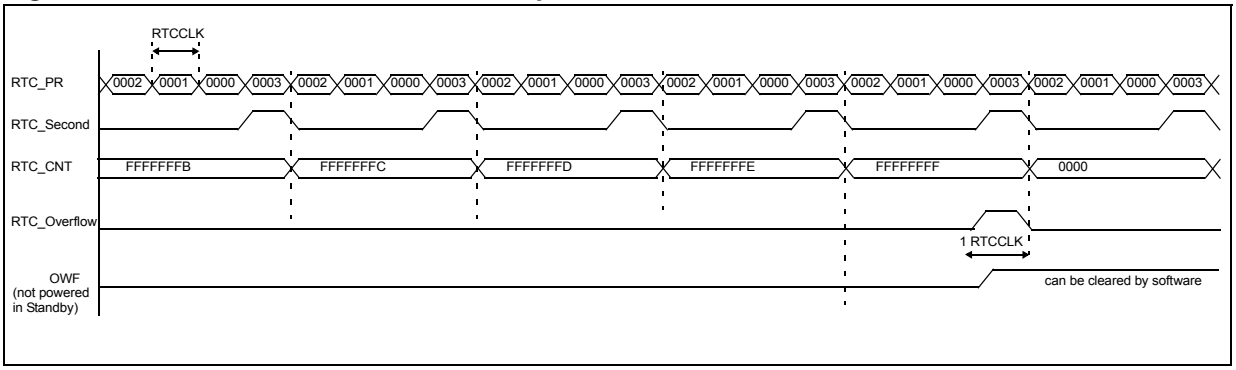


Figure 198. RTC Overflow waveform example with PR=0003



17.4 RTC registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

17.4.1 RTC control register high (RTC_CRH)

Address offset: 0x00
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													OWIE	ALRIE	SECIE
													rw	rw	rw

Bits 15:3 Reserved, forced by hardware to 0.

Bit 2 **OWIE:** Overflow interrupt enable
0: Overflow interrupt is masked.
1: Overflow interrupt is enabled.

Bit 1 **ALRIE**: Alarm interrupt enable
0: Alarm interrupt is masked.
1: Alarm interrupt is enabled.

Bit 0 **SECIE:** Second interrupt enable
0: Second interrupt is masked.
1: Second interrupt is enabled.

These bits are used to mask interrupt requests. Note that at reset all interrupts are disabled, so it is possible to write to the RTC registers to ensure that no interrupt requests are pending after initialization. It is not possible to write to the RTC_CRH register when the peripheral is completing a previous write operation (flagged by RTOFF=0, see *Section 17.3.4 on page 455*).

The RTC functions are controlled by this control register. Some bits must be written using a specific configuration procedure (see *Configuration procedure:*).

17.4.2 RTC control register low (RTC_CRL)

Address offset: 0x04
Reset value: 0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:6 Reserved, forced by hardware to 0.

Bit 5 **RTOFF:** RTC operation OFF

With this bit the RTC reports the status of the last write operation performed on its registers, indicating if it has been completed or not. If its value is '0' then it is not possible to write to any of the RTC registers. This bit is read only.
0: Last write operation on RTC registers is still ongoing.
1: Last write operation on RTC registers terminated.

Bit 4 **CNF**: Configuration flag

This bit must be set by software to enter in configuration mode so as to allow new values to be written in the RTC_CNT, RTC_ALR or RTC_PRL registers. The write operation is only executed when the CNF bit is reset by software after has been set.
0: Exit configuration mode (start update of RTC registers).
1: Enter configuration mode.

Bit 3 **RSF:** Registers synchronized flag

This bit is set by hardware at each time the RTC_CNT and RTC_DIV registers are updated and cleared by software. Before any read operation after an APB1 reset or an APB1 clock stop, this bit must be cleared by software, and the user application must wait until it is set to be sure that the RTC_CNT, RTC_ALR or RTC_PRL registers are synchronized.
0: Registers not yet synchronized.
1: Registers synchronized.

Bit 2 **OWF:** Overflow flag

This bit is set by hardware when the 32-bit programmable counter overflows. An interrupt is generated if OWIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.
0: Overflow not detected
1: 32-bit programmable counter overflow occurred.

Bit 1 **ALRF:** Alarm flag

This bit is set by hardware when the 32-bit programmable counter reaches the threshold set in the RTC_ALR register. An interrupt is generated if ALRIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.
0: Alarm not detected
1: Alarm detected

Bit 0 **SECF:** Second flag

This bit is set by hardware when the 32-bit programmable prescaler overflows, thus incrementing the RTC counter. Hence this flag provides a periodic signal with a period corresponding to the resolution programmed for the RTC counter (usually one second). An interrupt is generated if SECIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.
0: Second flag condition not met.
1: Second flag condition met.

The functions of the RTC are controlled by this control register. It is not possible to write to the RTC_CR register while the peripheral is completing a previous write operation (flagged by RTOFF=0, see *Section 17.3.4 on page 455*).

Note: 1 Any flag remains pending until the appropriate RTC_CR request bit is reset by software, indicating that the interrupt request has been granted.

2 At reset the interrupts are disabled, no interrupt requests are pending and it is possible to write to the RTC registers.

3 The OWF, ALRF, SECF and RSF bits are not updated when the APB1 clock is not running.

4 The OWF, ALRF, SECF and RSF bits can only be set by hardware and only cleared by software.

5 If ALRF = 1 and ALRIE = 1, the RTC global interrupt is enabled. If EXTI Line 17 is also enabled through the EXTI Controller, both the RTC global interrupt and the RTC Alarm interrupt are enabled.

6 If ALRF = 1, the RTC Alarm interrupt is enabled if EXTI Line 17 is enabled through the EXTI Controller in interrupt mode. When the EXTI Line 17 is enabled in event mode, a pulse is generated on this line (no RTC Alarm interrupt generation).

17.4.3 RTC prescaler load register (RTC_PRLH / RTC_PRLL)

The Prescaler Load registers keep the period counting value of the RTC prescaler. They are write-protected by the RTOFF bit in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'.

RTC prescaler load register high (RTC_PRLH)

Address offset: 0x08
Write only (see *Section 17.3.4 on page 455*)
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PRL[19:16]			
												w	w	w	w

Bits 15:4 Reserved, forced by hardware to 0.

Bits 3:0 **PRL[19:16]:** RTC prescaler reload value high
These bits are used to define the counter clock frequency according to the following formula:
$f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$

RTC prescaler load register low (RTC_PRLL)

Address offset: 0x0C
Write only (see *Section 17.3.4 on page 455*)
Reset value: 0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRL[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 15:0 **PRL[15:0]:** RTC prescaler reload value low
These bits are used to define the counter clock frequency according to the following formula:
$f_{TR\_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$

Caution: The zero value is not recommended. RTC interrupts and flags cannot be asserted correctly.

Note: If the input clock frequency (f_{RTCCLK}) is 32.768 kHz, write 7FFFh in this register to get a signal period of 1 second.

17.4.4 RTC prescaler divider register (RTC_DIVH / RTC_DIVL)

During each period of TR_CLK, the counter inside the RTC prescaler is reloaded with the value stored in the RTC_PRL register. To get an accurate time measurement it is possible to read the current value of the prescaler counter, stored in the RTC_DIV register, without stopping it. This register is read-only and it is reloaded by hardware after any change in the RTC_PRL or RTC_CNT registers.

RTC prescaler divider register high (RTC_DIVH)

Address offset: 0x10
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RTC_DIV[19:16]			
												r	r	r	r

Bits 15:4 Reserved

Bits 3:0 **RTC_DIV[19:16]:** RTC clock divider high

RTC prescaler divider register low (RTC_DIVL)

Address offset: 0x14
Reset value: 0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_DIV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **RTC_DIV[15:0]:** RTC clock divider low

17.4.5 RTC counter register (RTC_CNTH / RTC_CNTL)

The RTC core has one 32-bit programmable counter, accessed through two 16-bit registers; the count rate is based on the TR_CLK time reference, generated by the prescaler. RTC_CNT registers keep the counting value of this counter. They are write-protected by bit RTOFF in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'. A write operation on the upper (RTC_CNTH) or lower (RTC_CNTL) registers directly loads the corresponding programmable counter and reloads the RTC Prescaler. When reading, the current value in the counter (system date) is returned.

RTC counter register high (RTC_CNTH)

Address offset: 0x18
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **RTC_CNT[31:16]:** RTC counter high

Reading the RTC_CNTH register, the current value of the high part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode (see *Section 17.3.4: Configuring RTC registers on page 455*).

RTC counter register low (RTC_CNTL)

Address offset: 0x1C
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **RTC_CNT[15:0]:** RTC counter low

Reading the RTC_CNTL register, the current value of the lower part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode (see *Section 17.3.4: Configuring RTC registers on page 455*).

17.4.6 RTC alarm register high (RTC_ALRH / RTC_ALRL)

When the programmable counter reaches the 32-bit value stored in the RTC_ALR register, an alarm is triggered and the RTC_alarmIT interrupt request is generated. This register is write-protected by the RTOFF bit in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'.

RTC alarm register high (RTC_ALRH)

Address offset: 0x20
Write only (see *Section 17.3.4 on page 455*)
Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 15:0 **RTC_ALR[31:16]:** RTC alarm high

The high part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode (see *Section 17.3.4: Configuring RTC registers on page 455*).

RTC alarm register low (RTC_ALRL)

Address offset: 0x24
Write only (see *Section 17.3.4 on page 455*)
Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 15:0 **RTC_ALR[15:0]:** RTC alarm low

The low part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode (see *Section 17.3.4: Configuring RTC registers on page 455*).

17.4.7 RTC register map

RTC registers are mapped as 16-bit addressable registers as described in the table below:

Table 85. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_CRH**	Reserved																													OWIE	ALRIE	SECIE
	Reset value																														0	0	0
0x04	**RTC_CRL**	Reserved																										RTOFF	CNF	RSF	OWF	ALRF	SECF
	Reset value																											1	0	0	0	0	0
0x08	**RTC_PRLH**	Reserved																												PRL[19:16]			
	Reset value																													0	0	0	0
0x0C	**RTC_PRLL**	Reserved																PRL[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**RTC_DIVH**	Reserved																DIV[31:16]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**RTC_DIVL**	Reserved																DIV[15:0]															
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**RTC_CNTH**	Reserved																CNT[13:16]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**RTC_CNTL**	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**RTC_ALRH**	Reserved																ALR[31:16]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x24	**RTC_ALRL**	Reserved																ALR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

18 Independent watchdog (IWDG)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

18.1 IWDG introduction

The STM32F100xx have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window. For further information on the window watchdog, refer to *Section 19 on page 469*.

18.2 IWDG main features

- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

18.3 IWDG functional description

Figure 199 shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the 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 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

18.3.1 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and will generate a reset unless the Key register is written by the software before the counter reaches end of count.

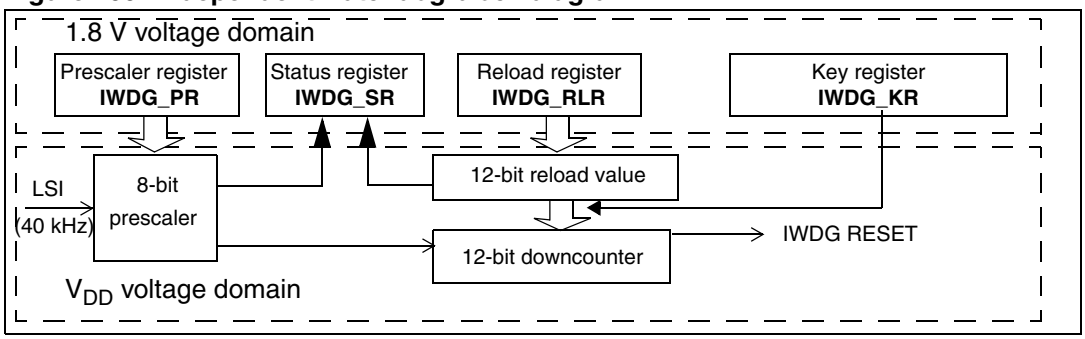
18.3.2 Register access protection

Write access to the IWDG_PR and IWDG_RLR registers is protected. To modify them, you must first write the code 0x5555 in the IWDG_KR register. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0xAAAA).
A status register is available to indicate that an update of the prescaler or the down-counter reload value is on going.

18.3.3 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module. For more details, refer to *Section 25.15.2: Debug support for timers, watchdog and I2C*.

Figure 199. Independent watchdog block diagram



Note: The watchdog function is implemented in the V_{DD} voltage domain that is still functional in Stop and Standby modes.

Table 86. Min/max IWDG timeout period at 40 kHz (LSI) (1)

Prescaler divider	PR[2:0] bits	Min timeout (ms) RL[11:0]= 0x000	Max timeout (ms) RL[11:0]= 0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	6 (or 7)	6.4	26214.4

1. These timings are given for a kHz clock but the microcontroller’s internal RC frequency can vary from 30 to 60 kHz. Moreover, given an exact RC oscillator frequency, the exact timings still depend on the phasing of the APB interface clock versus the LSI clock so that there is always a full RC period of uncertainty.

The LSI can be calibrated so as to compute the IWDG timeout with an acceptable accuracy. For more details refer to *LSI clock on page 93*.

18.4 IWDG registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

18.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

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
Reserved																KEY[15:0]															
																w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, read as 0.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0000h)

These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see *Section 18.3.2*)

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

18.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													PR[2:0]		
																													rw	rw	rw

Bits 31:3 Reserved, read as 0.

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected see *Section 18.3.2*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of 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_SR register is reset.

18.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

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
Reserved																				RL[11:0]											
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, read as 0.

Bits11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see *Section 18.3.2*. They are written by software to define the value to be loaded in the watchdog counter each time the value AAAAh is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to *Table 86.*

The RVU bit in the IWDG_SR register 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 up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

18.4.4 Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

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
Reserved																														RVU	PVU
																														r	r

Bits 31:2 Reserved

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_{DD} voltage domain (takes up to 5 RC 40 kHz 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_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: If several reload values or prescaler values are used by application, it is mandatory to wait until RVU bit is reset before changing the reload value and to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler and/or the reload value it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete)

18.4.5 IWDG register map

The following table gives the IWDG register map and reset values.

Table 87. IWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Reserved																														PR[2:0]	
	Reset value																															0	0	0
0x08	**IWDG_RLR**	Reserved																RL[11:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Reserved																														RVU	PVU
	Reset value																															0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

19 Window watchdog (WWDG)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

19.1 WWDG introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

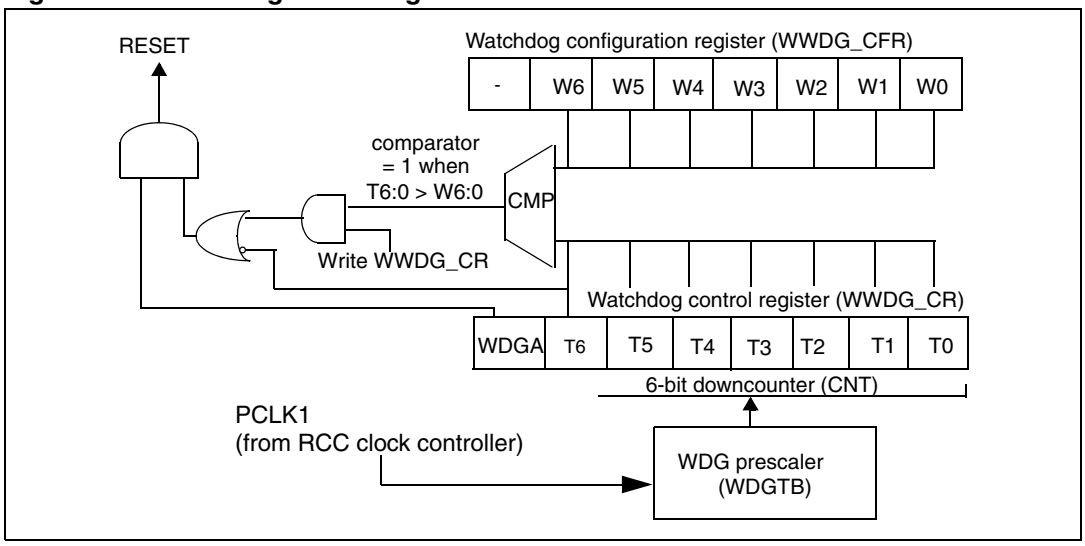
19.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see *Figure 201*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

19.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over 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.

Figure 200. Watchdog block diagram



The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0:

Enabling the 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.

Controlling the downcounter

This downcounter is free-running: It counts down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 201*).The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. *Figure 201* 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).

Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this

case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

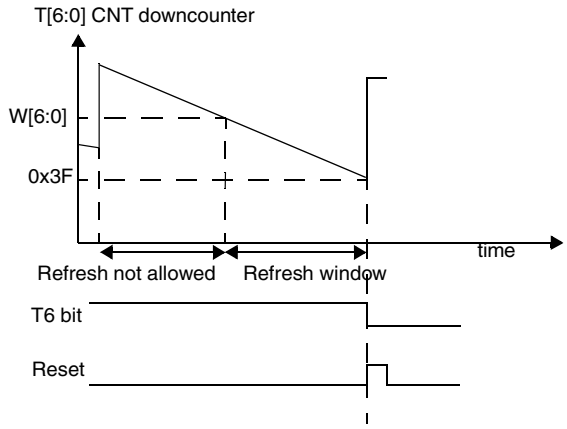
Note: When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.

19.4 How to program the watchdog timeout

You can use the formula in *Figure 201* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 201. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$T_{WWDG} = T_{PCLK1} \times 4096 \times 2^{WDGTB} \times (T[5:0] + 1) \quad \text{(in ms)}$$

where:

T_{WWDG}: WWDG timeout

T_{PCLK1}: APB1 clock period measured in ms

Min-max timeout value at 24 MHz (PCLK1)

Prescaler	WDGTB	Min timeout (µs) T[5:0] = 0x00	Max timeout (ms) T[5:0] = 0x3F
1	0	170.67	10.92
2	1	341.33	21.85
4	2	682.67	43.69
8	3	1365.33	87.38

ai17308

19.5 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module. For more details, refer to *Section 25.15.2: Debug support for timers, watchdog and I2C*.

19.6 WWDG registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

19.6.1 Control register (WWDG_CR)

Address offset: 0x00

Reset value: 0x7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T[6:0]						
								rs	rw						

Bits 31:8 Reserved

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter. It is decremented every (4096 x 2^{WDGTB}) PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

19.6.2 Configuration register (WWDG_CFR)

Address offset: 0x04

Reset value: 0x7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB[1:0]		W[6:0]						
						rs	rw		rw						

Bit 31:10 Reserved

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 **WDGTB[1:0]:** Timer base

The time base of the prescaler can be modified as follows:
00: CK Counter Clock (PCLK1 div 4096) div 1
01: CK Counter Clock (PCLK1 div 4096) div 2
10: CK Counter Clock (PCLK1 div 4096) div 4
11: CK Counter Clock (PCLK1 div 4096) div 8

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared to the downcounter.

19.6.3 Status register (WWDG_SR)

Address offset: 0x08
Reset value: 0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

Bit 31:1 Reserved

Bit 0 **EWIF:** Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0. A write of ‘1 has no effect. This bit is also set if the interrupt is not enabled.

19.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 88. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**WWDG_CR**	Reserved																								WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	**WWDG_CFR**	Reserved																						EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	**WWDG_SR**	Reserved																															EWIF
	Reset value																																0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

20 Flexible static memory controller (FSMC)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to high-density value line devices only.

20.1 FSMC main features

The FSMC block is able to interface with synchronous and asynchronous memories. Its main purpose is to:

- Translate the AHB transactions into the appropriate external device protocol
- Meet the access timing requirements of the external devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique chip select. The FSMC performs only one access at a time to an external device.

The FSMC has the following main features:

- Interfaces with static memory-mapped devices including:
  - Static random access memory (SRAM)
  - Read-only memory (ROM)
  - NOR Flash memory/OneNAND Flash memory
  - PSRAM (4 memory banks)
- Supports burst mode access to synchronous devices (NOR Flash and PSRAM)
- 8- or 16-bit wide databus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Programmable timings to support a wide range of devices, in particular:
  - Programmable wait states (up to 15)
  - Programmable bus turnaround cycles (up to 15)
  - Programmable output enable and write enable delays (up to 15)
  - Independent read and write timings and protocol, so as to support the widest variety of memories and timings
- Write enable and byte lane select outputs for use with PSRAM and SRAM devices
- Translation of 32-bit wide AHB transactions into consecutive 16-bit or 8-bit accesses to external 16-bit or 8-bit devices
- A Write FIFO, 2 words long, each word is 32 bits wide, only stores data and not the address. Therefore, this FIFO only buffers AHB write burst transactions. This makes it possible to write to slow memories and free the AHB quickly for other operations. Only one burst at a time is buffered: if a new AHB burst or single transaction occurs while an

operation is in progress, the FIFO is drained. The FSMC will insert wait states until the current memory access is complete).

- External asynchronous wait control

The FSMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up. However, it is possible to change the settings at any time.

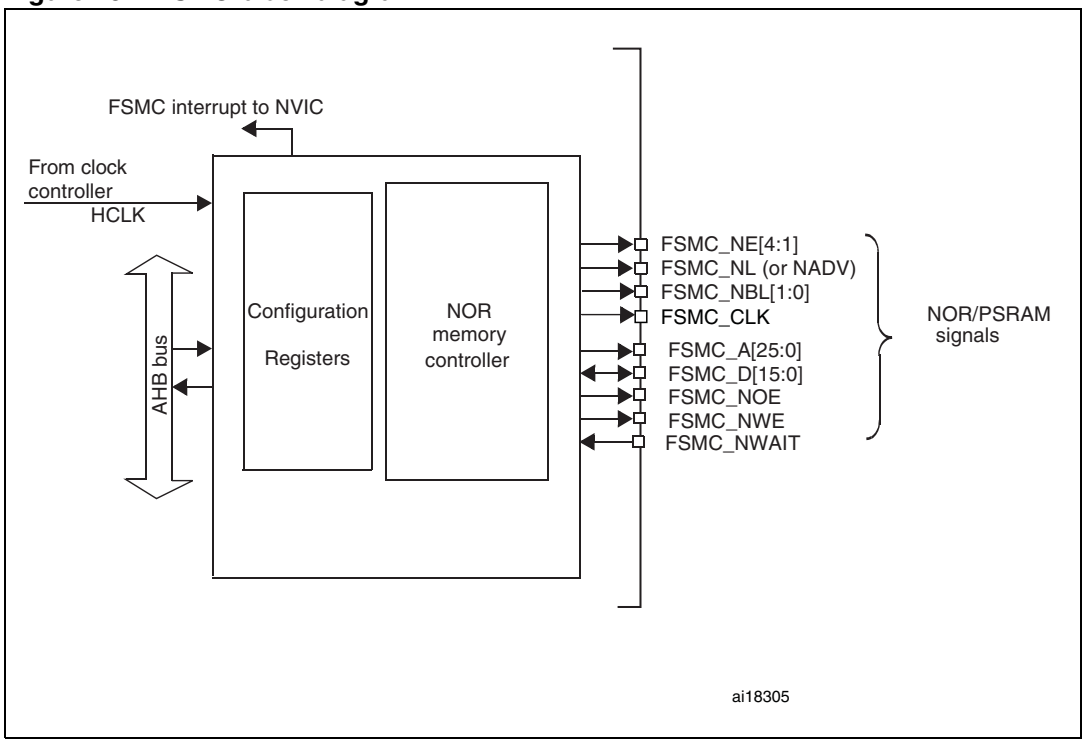
20.2 Block diagram

The FSMC consists of four main blocks:

- The AHB interface (including the FSMC configuration registers)
- The NOR Flash/PSRAM controller
- The external device interface

The block diagram is shown in *Figure 202*.

Figure 202. FSMC block diagram



20.3 AHB interface

The AHB slave interface enables internal CPUs and other bus master peripherals to access the external static memories.

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16 or 8 bits wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The Chip Select toggles for each access.

The FSMC generates an AHB error in the following conditions:

- When reading or writing to an FSMC bank which is not enabled
- When reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FSMC_BCRx register.

The effect of this AHB error depends on the AHB master which has attempted the R/W access:

- If it is the Cortex™-M3 CPU, a hard fault interrupt is generated
- If is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

The AHB clock (HCLK) is the reference clock for the FSMC.

20.3.1 Supported memories and transactions

General transaction rules

The requested AHB transaction data size can be 8-, 16- or 32-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal
  There is no issue in this case.
- AHB transaction size is greater than the memory size
  In this case, the FSMC splits the AHB transaction into smaller consecutive memory accesses in order to meet the external data width.
- AHB transaction size is smaller than the memory size
  Asynchronous transfers may or not be consistent depending on the type of external device.
  - Asynchronous accesses to devices that have the byte select feature (SRAM, ROM, PSRAM).

    a) FSMC allows write transactions accessing the right data through its byte lanes NBL[1:0]

    b) Read transactions are allowed (the controller reads the entire memory word and uses the needed byte only). The NBL[1:0] are always kept low during read transactions.
  - Asynchronous accesses to devices that do not have the byte select feature (NOR).
    This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Clearly, the device cannot be accessed in byte mode (only 16-bit words can be read from/written to the Flash memory) therefore:

  a) Write transactions are not allowed

  b) Read transactions are allowed (the controller reads the entire 16-bit memory word and uses the needed byte only).

Configuration registers

The FSMC can be configured using a register set. See *Section 20.5.6*, for a detailed description of the NOR Flash/PSRAM controller registers.

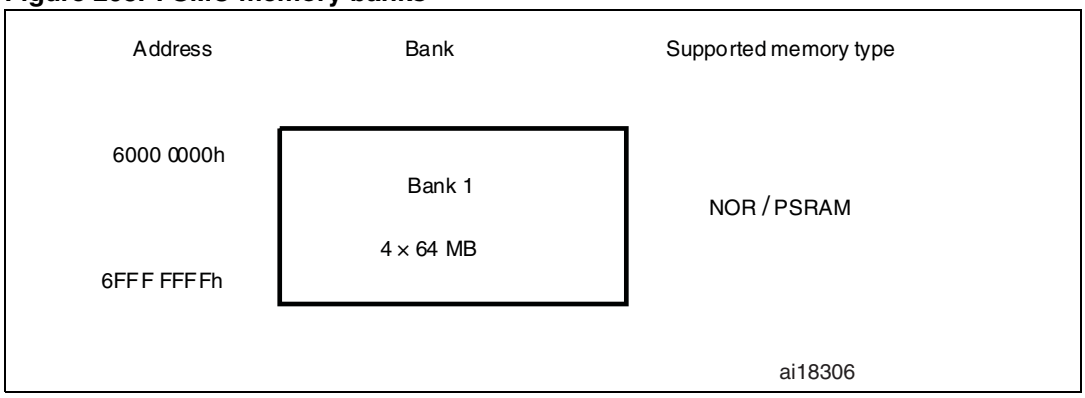
20.4 External device address mapping

From the FSMC point of view, the external memory is composed of a single fixed size bank of 256 Mbytes (Refer to *Figure 203*):

- Bank 1 used to address up to 4 NOR Flash or PSRAM memory devices. This bank is split into 4 NOR/PSRAM regions with 4 dedicated Chip Select.

For each bank the type of memory to be used is user-defined in the Configuration register.

Figure 203. FSMC memory banks



20.4.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 89*.

Table 89. NOR/PSRAM bank selection

HADDR[27:26](1)	Selected bank
00	Bank 1 NOR/PSRAM 1
01	Bank 1 NOR/PSRAM 2
10	Bank 1 NOR/PSRAM 3
11	Bank 1 NOR/PSRAM 4

1. HADDR are internal AHB address lines that are translated to external memory.

HADDR[25:0] contain the external memory address. Since HADDR is a byte address whereas the memory is addressed in words, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 90. External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit

1. In case of a 16-bit external memory width, the FSMC will internally use HADDR[25:1] to generate the address for external memory FSMC_A[24:0].
Whatever the external memory width (16-bit or 8-bit), FSMC_A[0] should be connected to external memory address A[0].

Wrap support for NOR Flash/PSRAM

Wrap burst mode for synchronous memories is not supported. The memories must be configured in linear burst mode of undefined length.

20.5 NOR Flash/PSRAM controller

The FSMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
  - 8-bit
  - 16-bit
  - 32-bit
- PSRAM (Cellular RAM)
  - Asynchronous mode
  - Burst mode
- NOR Flash
  - Asynchronous mode or burst mode
  - Multiplexed or nonmultiplexed

The FSMC outputs a unique chip select signal NE[4:1] per bank. All the other signals (addresses, data and control) are shared.

For synchronous accesses, the FSMC issues the clock (CLK) to the selected external device. This clock is a submultiple of the HCLK clock. The size of each bank is fixed and equal to 64 Mbytes.

Each bank is configured by means of dedicated registers (see *Section 20.5.6*).

The programmable memory parameters include access timings (see *Table 91*) and support for wait management (for PSRAM and NOR Flash accessed in burst mode).

Table 91. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	1	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

20.5.1 External memory interface signals

Table 92, *Table 93* and *Table 94* list the signals that are typically used to interface NOR Flash, SRAM and PSRAM.

Note: Prefix "N". specifies the associated signal as active low.

NOR Flash, nonmultiplexed I/Os

Table 92. Nonmultipled I/O NOR Flash

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FSMC

NOR Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

NOR Flash, multiplexed I/Os

Table 93. Multiplexed I/O NOR Flash

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FSMC

NOR-Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM/SRAM, nonmultiplexed I/Os

Table 94. Nonmultiplexed I/Os PSRAM/SRAM

FSMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous burst)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lowed byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM, multiplexed I/Os

Table 95. Multiplexed I/O PSRAM

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lowed byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

20.5.2 Supported memories and transactions

Table 96 below displays an example of the supported devices, access modes and transactions when the memory data bus is 16-bit for NOR, PSRAM and SRAM. Transactions not allowed (or not supported) by the FSMC in this example appear in gray.

Table 96. NOR Flash/PSRAM supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NOR Flash (muxed I/Os and nonmuxed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	
PSRAM (multiplexed I/Os and nonmultiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32	16	Y	

Table 96. NOR Flash/PSRAM supported memories and transactions (continued)

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses

20.5.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In synchronous read and write mode, the output data changes on the falling edge of the memory clock (FSMC_CLK).

20.5.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR Flash, SRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FSMC always samples the data before de-asserting the chip select signal NE. This guarantees that the memory data-hold timing constraint is met (chip enable high to data transition, usually 0 ns min.)
- When extended mode is set, it is possible to mix modes A, B, C and D in read and write (it is for instance possible to read in mode A and write in mode B).

Mode 1 - SRAM/CRAM

Figure 204. Mode1 read accesses

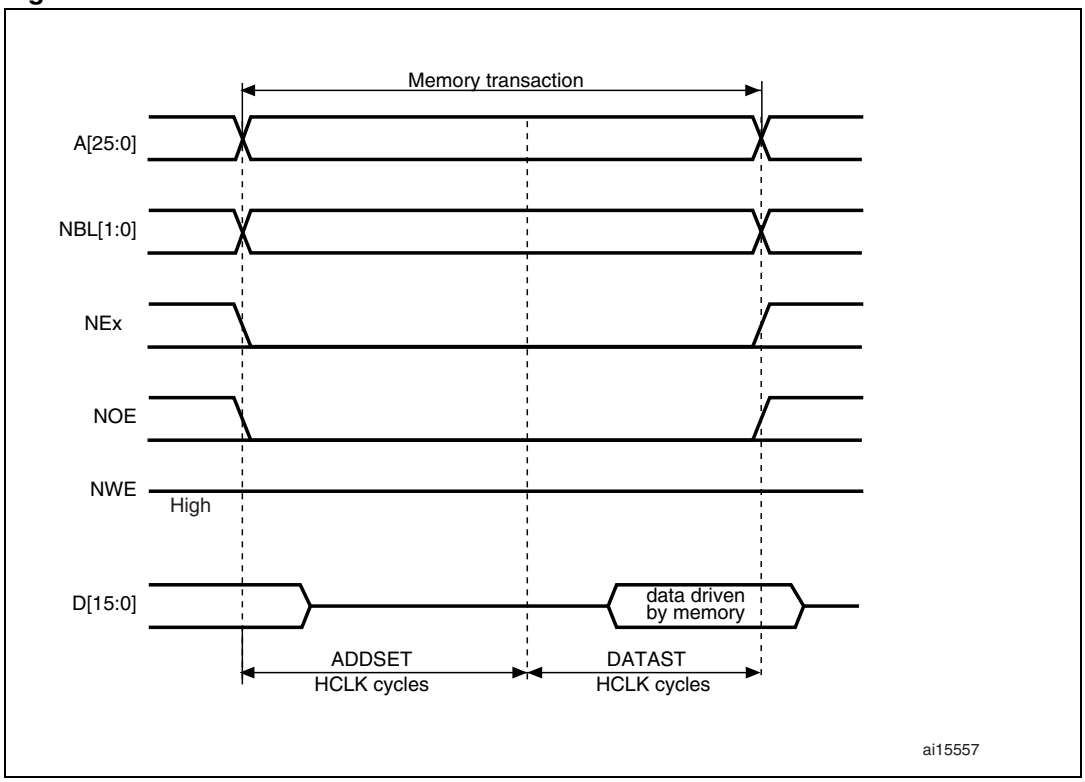
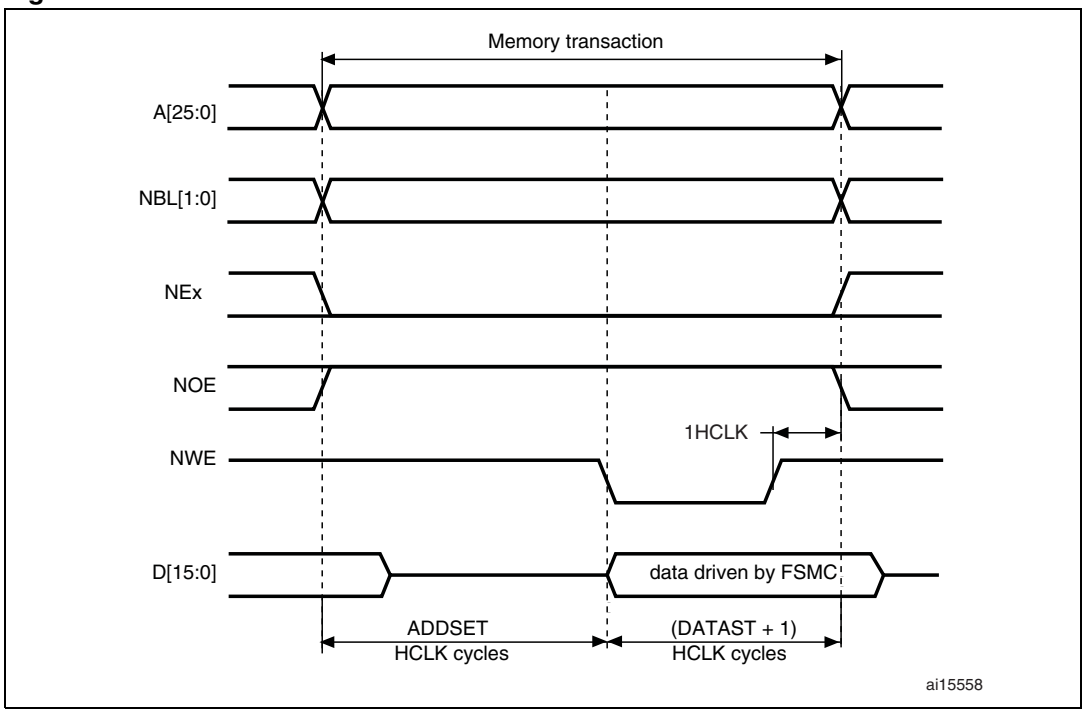


Figure 205. Mode1 write accesses



1. NBL[1:0] are driven low during read access.

The one HCLK cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this one HCLK cycle, the DATAST value must be greater than zero (DATAST > 0).

Table 97. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 98. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-20		0x0000
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles for write accesses, DATAST HCLK cycles for read accesses).
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.

Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 206. ModeA read accesses

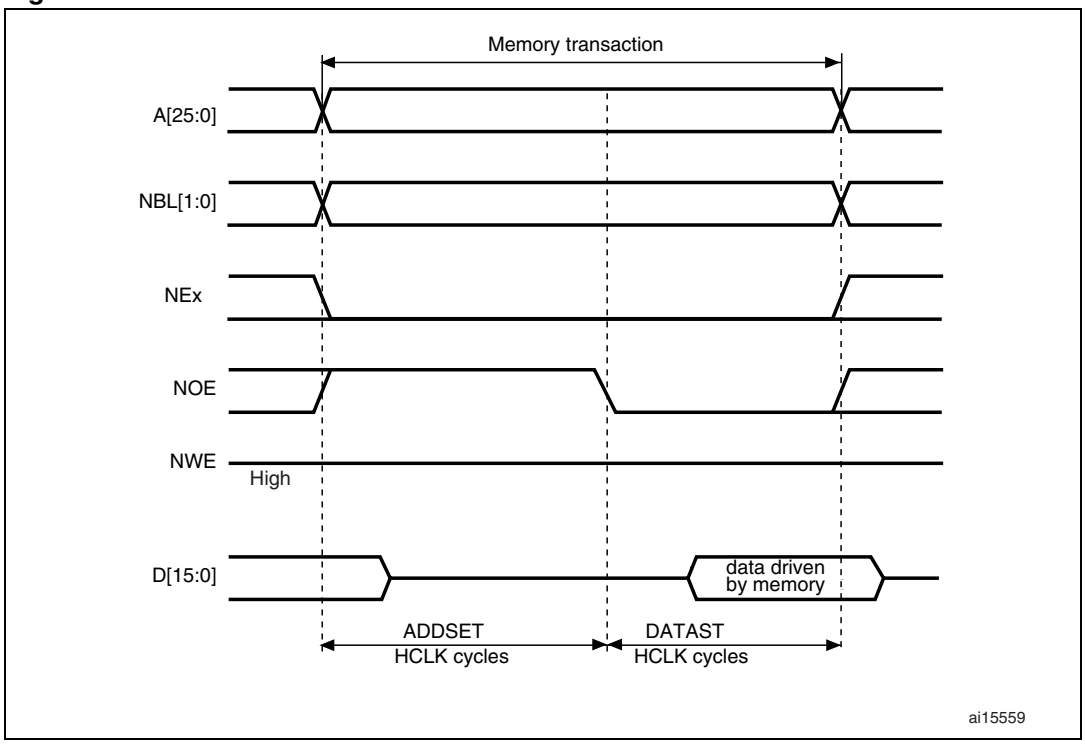
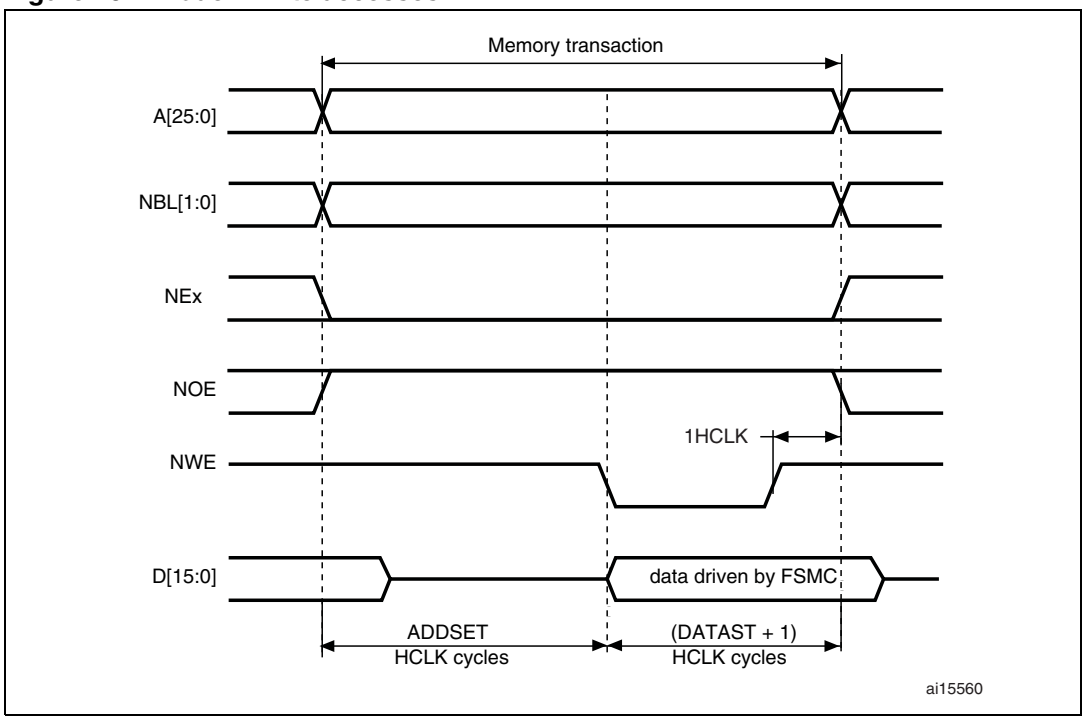


Figure 207. ModeA write accesses



1. NBL[1:0] are driven low during read access.

The differences compared with mode1 are the toggling of NOE and the independent read and write timings.

Table 99. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 100. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x0
27-20		0x000
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 101. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x0
27-20		0x000
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles) in write.

Table 101. FSMC_BWTRx bit fields (continued)

Bit number	Bit name	Value to set
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write Minimum value for ADDSET is 1.

Mode 2/B - NOR Flash

Figure 208. Mode2/B read accesses

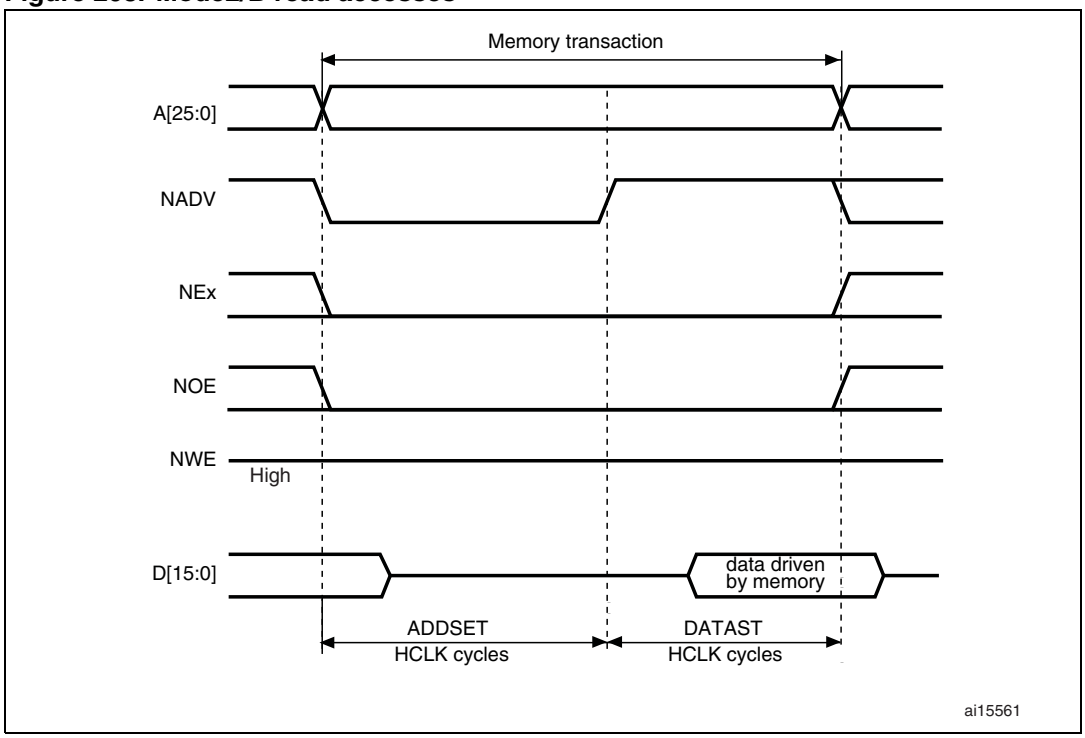


Figure 209. Mode2 write accesses

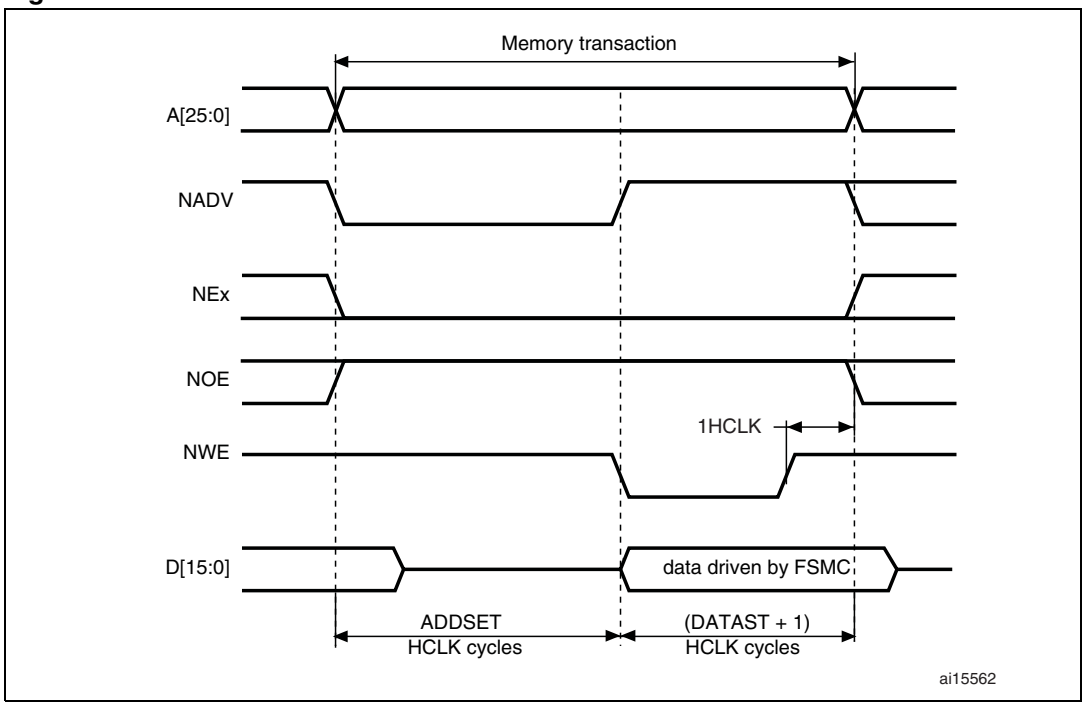
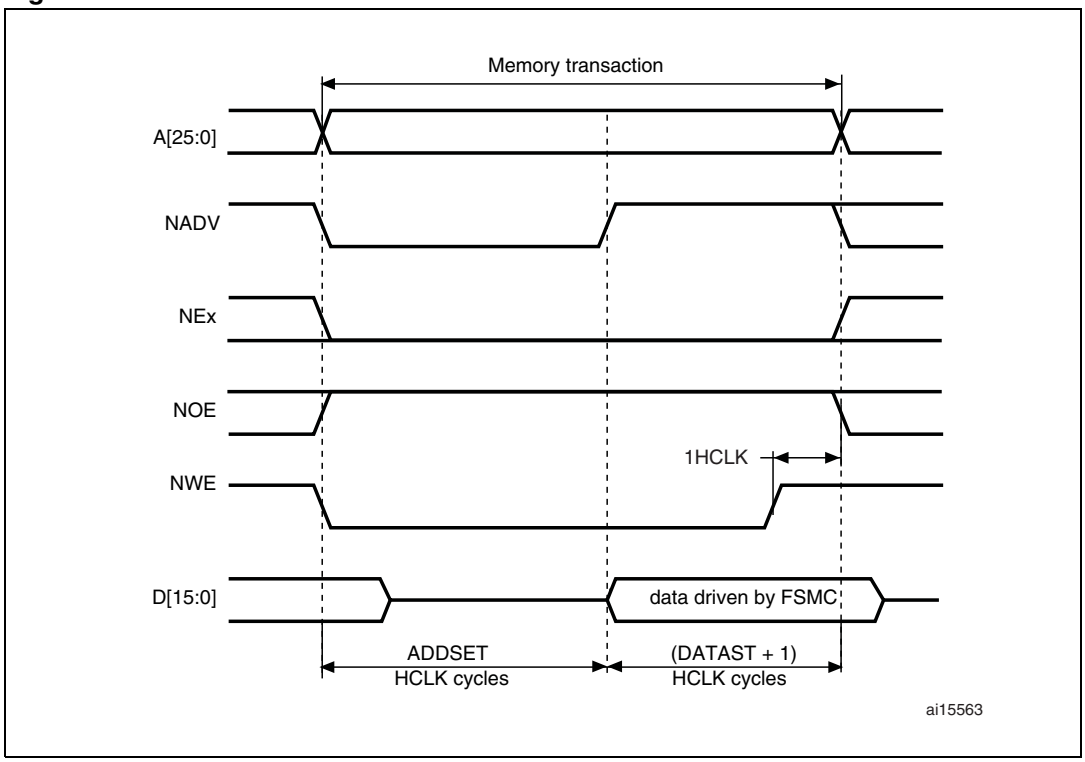


Figure 210. ModeB write accesses



The differences with mode1 are the toggling of NADV and the independent read and write timings when extended mode is set (Mode B).

Table 102. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 103. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x1 if extended mode is set
27-20		0x000
15-8	DATAST	Duration of the access second phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 104. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x1 if extended mode is set
27-20		0x000
15-8	DATAST	Duration of the access second phase (DATAST+1 HCLK cycles) in write.
7-4		0x0
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

Note: The FSMC_BWTRx register is valid only if extended mode is set (mode B), otherwise all its content is don't care.

Mode C - NOR Flash - OE toggling

Figure 211. ModeC read accesses

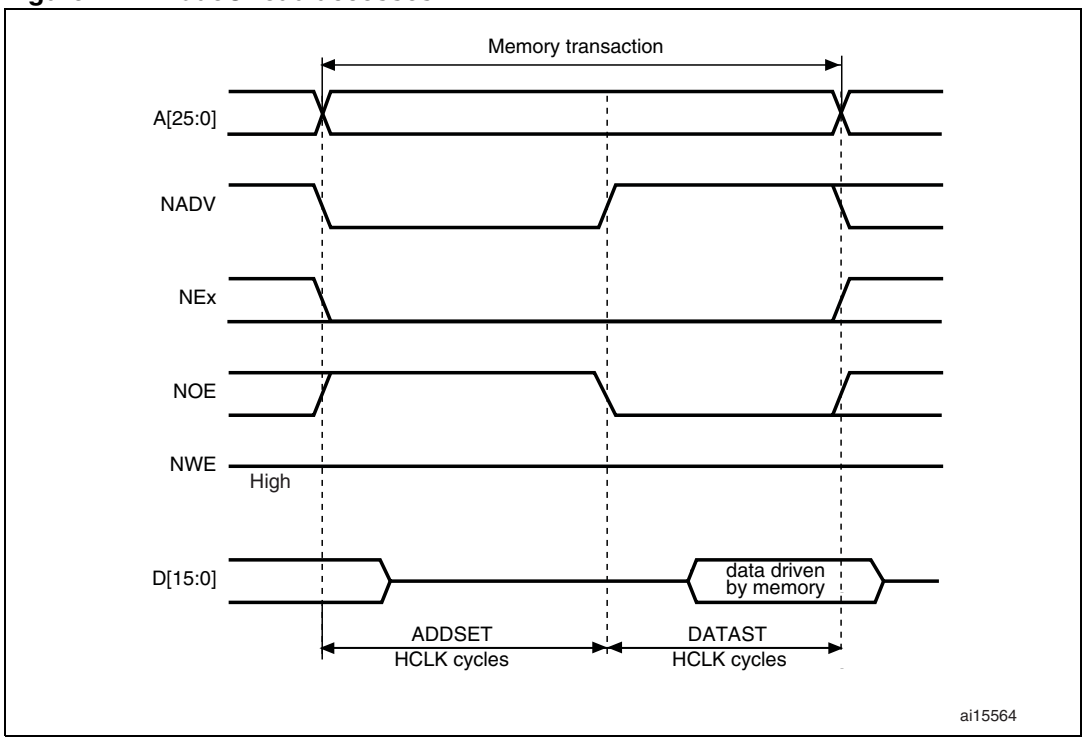
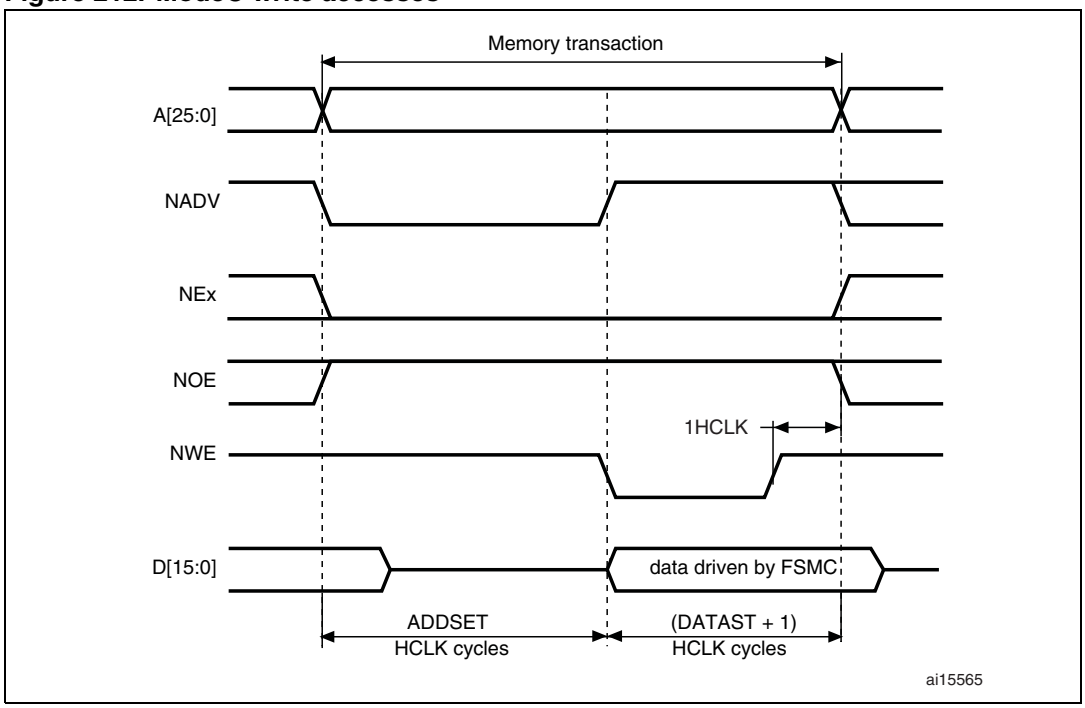


Figure 212. ModeC write accesses



The differences compared with mode1 are the toggling of NOE and NADV and the independent read and write timings.

Table 105. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	1
5-4	MWID	As needed
3-2	MTYP	0x02 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 106. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSETHCLK cycles) in read. Minimum value for ADDSET is 1.

Table 107. FSMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles) in write.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

Mode D - asynchronous access with extended address

Figure 213. ModeD read accesses

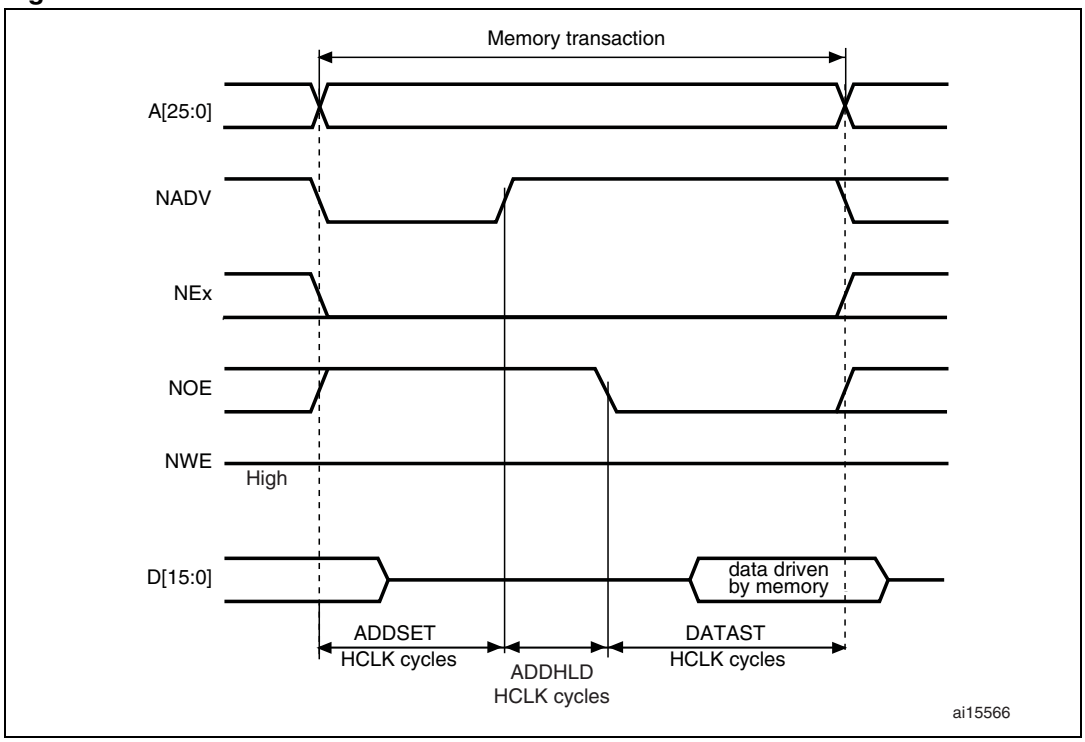
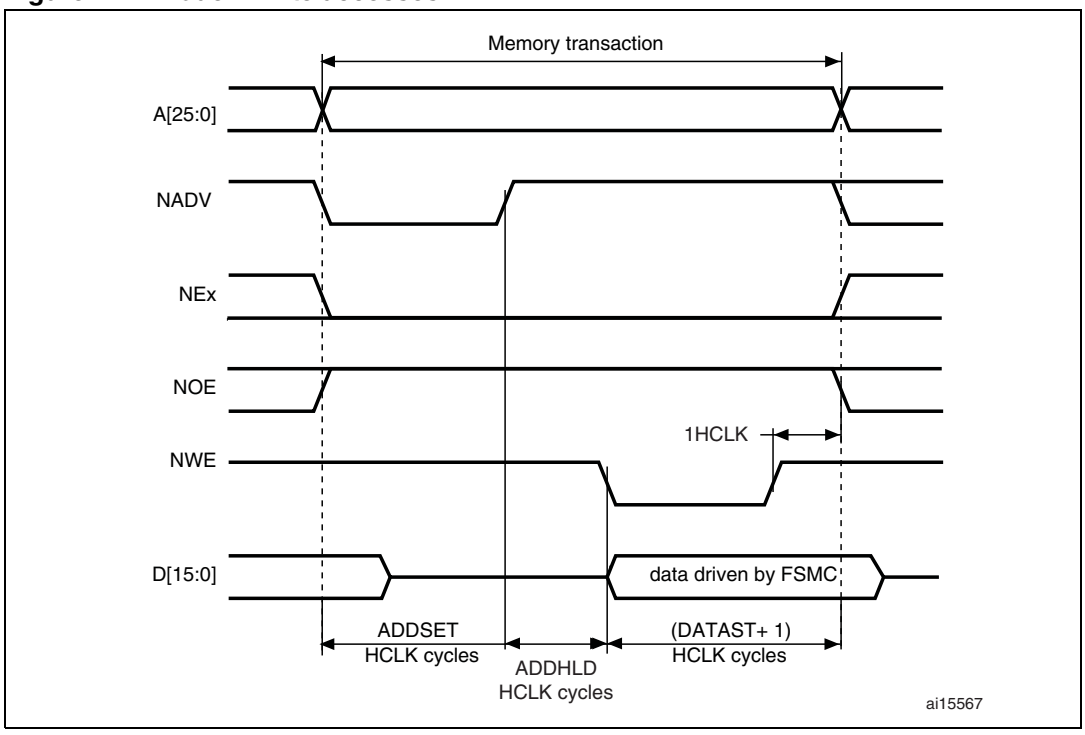


Figure 214. ModeD write accesses



The differences with mode1 are the toggling of NADV, NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 108. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 109. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSETHCLK cycles) in read. Minimum value for ADDSET is 1.

Table 110. FSMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles) in write.

Table 110. FSMC_BWTRx bit fields (continued)

Bit No.	Bit name	Value to set
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

Mode muxed - asynchronous access muxed NOR Flash

Figure 215. Multiplexed read accesses

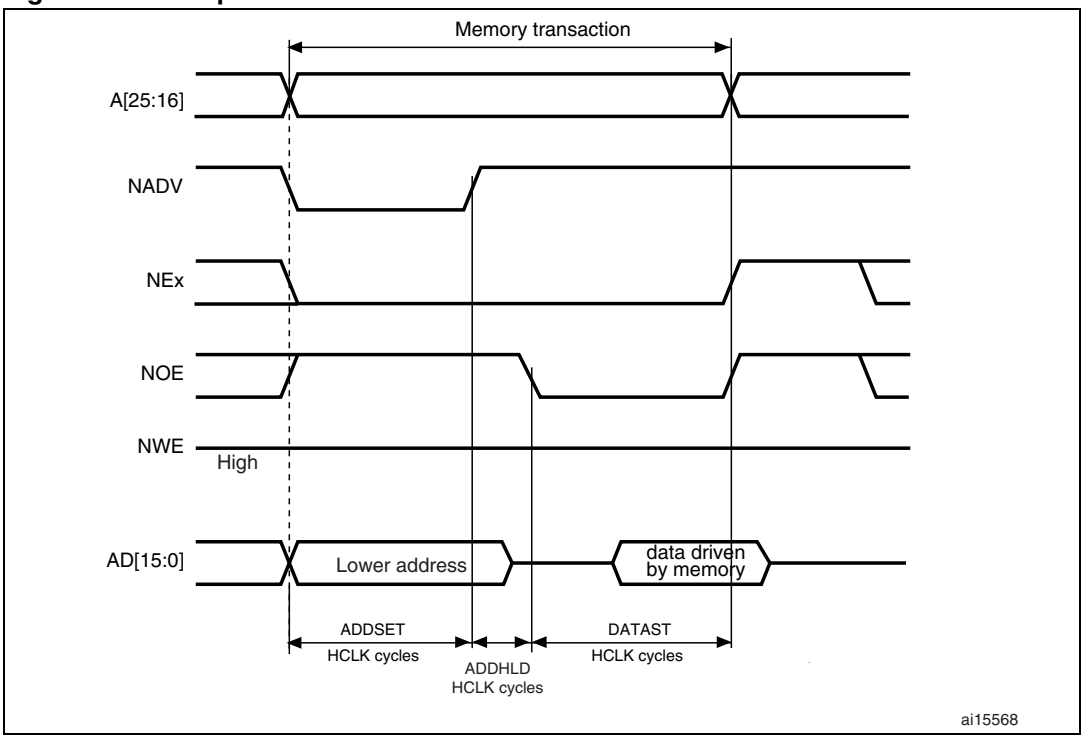
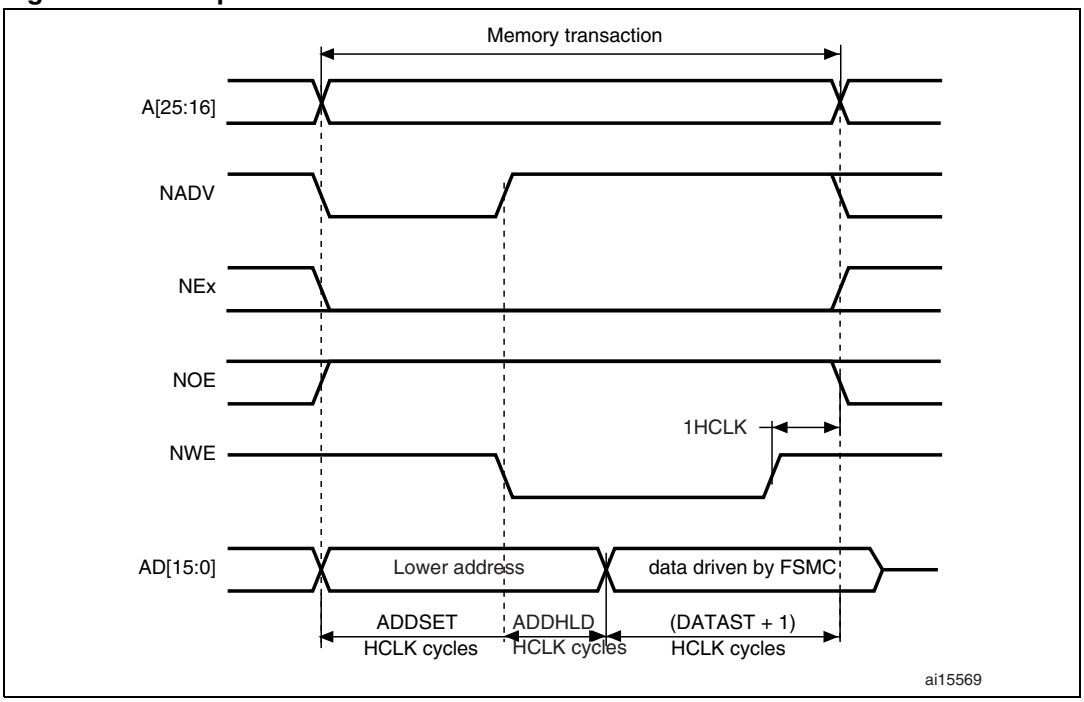


Figure 216. Multiplexed write accesses



The difference with mode D is the drive of the lower address byte(s) on the databus.

Table 111. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR)
1	MUXEN	0x1
0	MBKEN	0x1

Table 112. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)

Table 112. FSMC_BTRx bit fields (continued)

Bit No.	Bit name	Value to set
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles for read accesses and DATAST+1 HCLK cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD HCLK cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts a WAIT signal to advise that it's not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FSMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase) programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data phase must be programmed so that WAIT can be detected 4 HCLK cycles before the data sampling. The following cases must be considered:

1. Memory asserts the WAIT signal aligned to NOE/NWE which toggles:

   data_setup phase >= 4 * HCLK + max_wait_assertion_time

2. Memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):

   if max_wait_assertion_time > (address_phase + hold_phase)

   data_setup phase >= 4 * HCLK + (max_wait_assertion_time - address_phase - hold_phase)

   otherwise

   data_setup phase >= 4 * HCLK

Where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

The *Figure 217* and *Figure 218* show the number of HCLK clock cycles that memory access is extended after WAIT is removed by the asynchronous memory (independently of the above cases).

Figure 217. Asynchronous wait during a read access

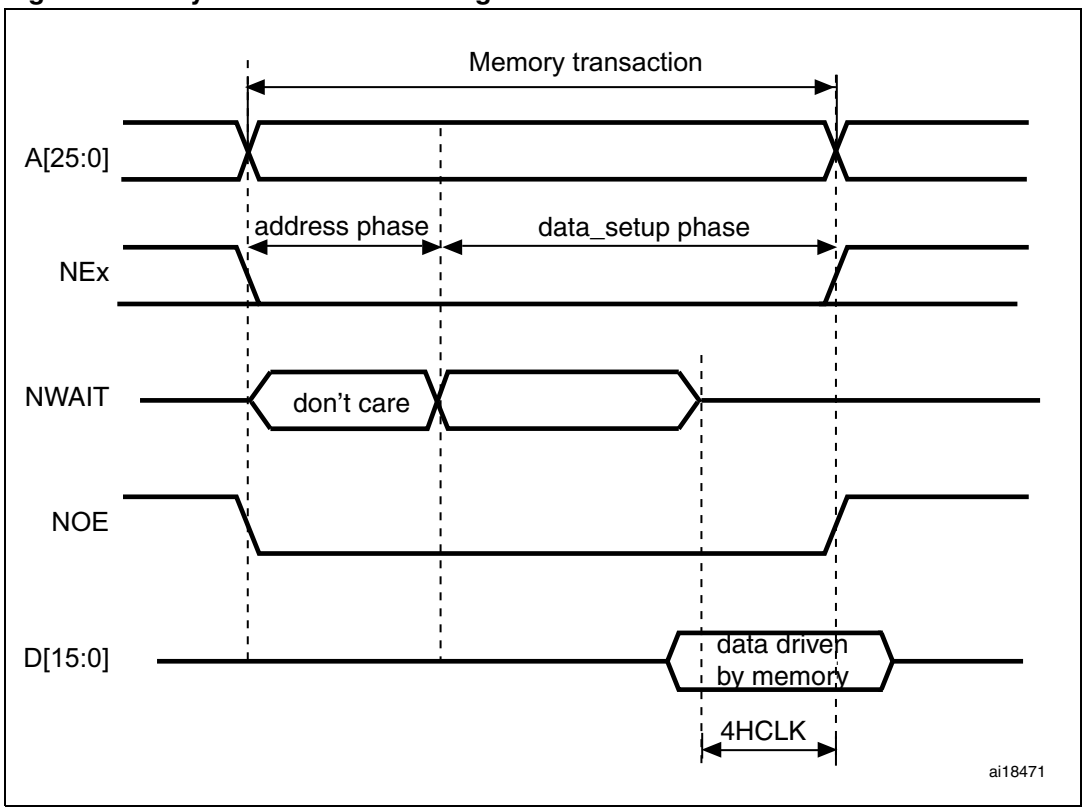
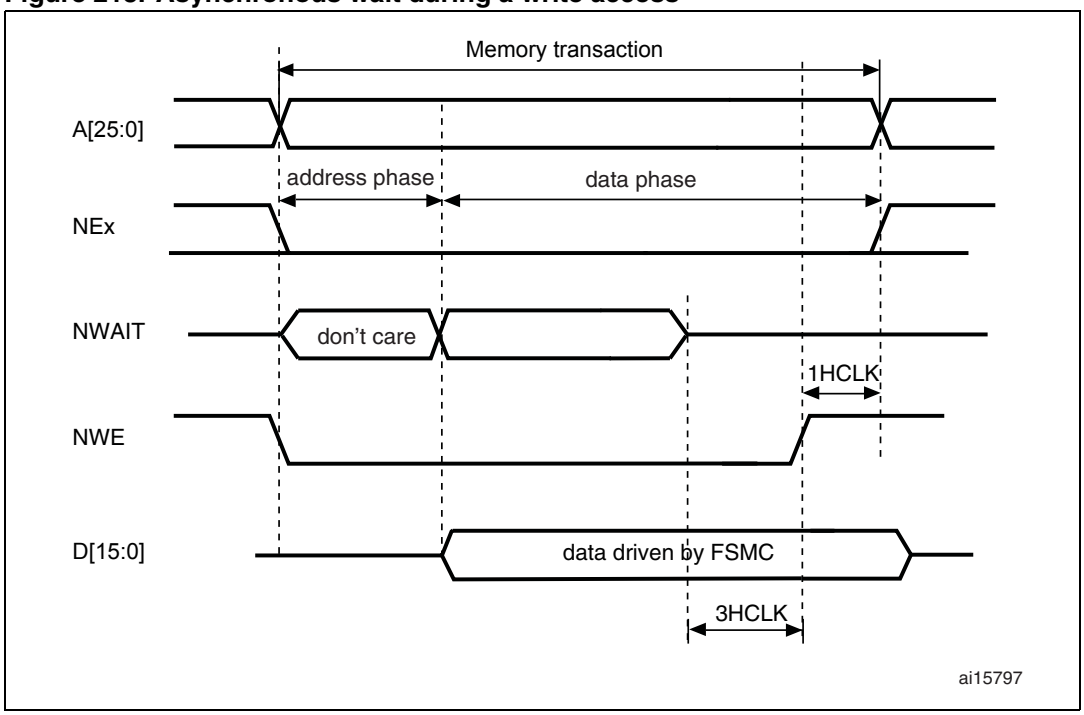


Figure 218. Asynchronous wait during a write access



20.5.5 Synchronous burst transactions

The memory clock, CLK, is a submultiple of HCLK according to the value of parameter CLKDIV.

NOR Flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, the FSMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs *in the middle* of the NADV low pulse.

Data latency versus NOR Flash latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR Flash configuration register. The FSMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR Flash memories include the NADV Low cycle in the data latency count, so the exact relation between the NOR Flash latency and the FMSC DATLAT parameter can be either of:

- NOR Flash latency = DATLAT + 2
- NOR Flash latency = DATLAT + 3

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FSMC samples the data and waits long enough to evaluate if the data are valid. Thus the FSMC detects when the memory exits latency and real data are taken.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FSMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in synchronous burst mode, if an AHB single-burst transaction is requested, the FSMC performs a burst transaction of length 1 (if the AHB transfer is 16-bit), or length 2 (if the AHB transfer is 32-bit) and de-assert the chip select signal when the last data is strobed.

Clearly, such a transfer is not the most efficient in terms of cycles (compared to an asynchronous read). Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Wait management

For synchronous burst NOR Flash, NWAIT is evaluated after the programmed latency period, (DATALAT+2) CLK clock cycles.

If NWAIT is sensed active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is sensed inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the chip select and output enable signals valid, and does not consider the data valid.

There are two timing configurations for the NOR Flash NWAIT signal in burst mode:

- Flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset)
- Flash memory asserts the NWAIT signal during the wait state

These two NOR Flash wait state configurations are supported by the FSMC, individually for each chip select, thanks to the WAITCFG bit in the FSMC_BCRx registers (x = 0..3).

Figure 219. Wait configurations

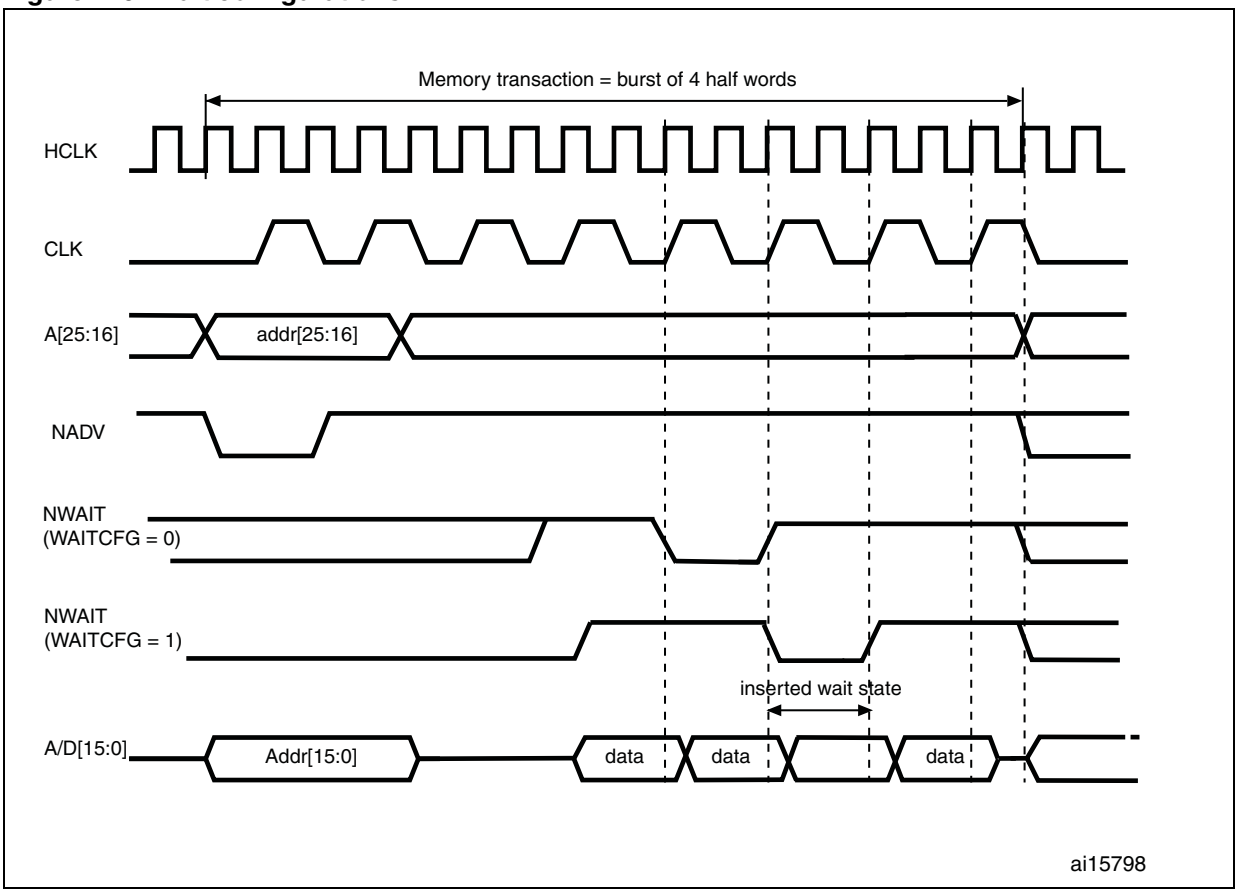
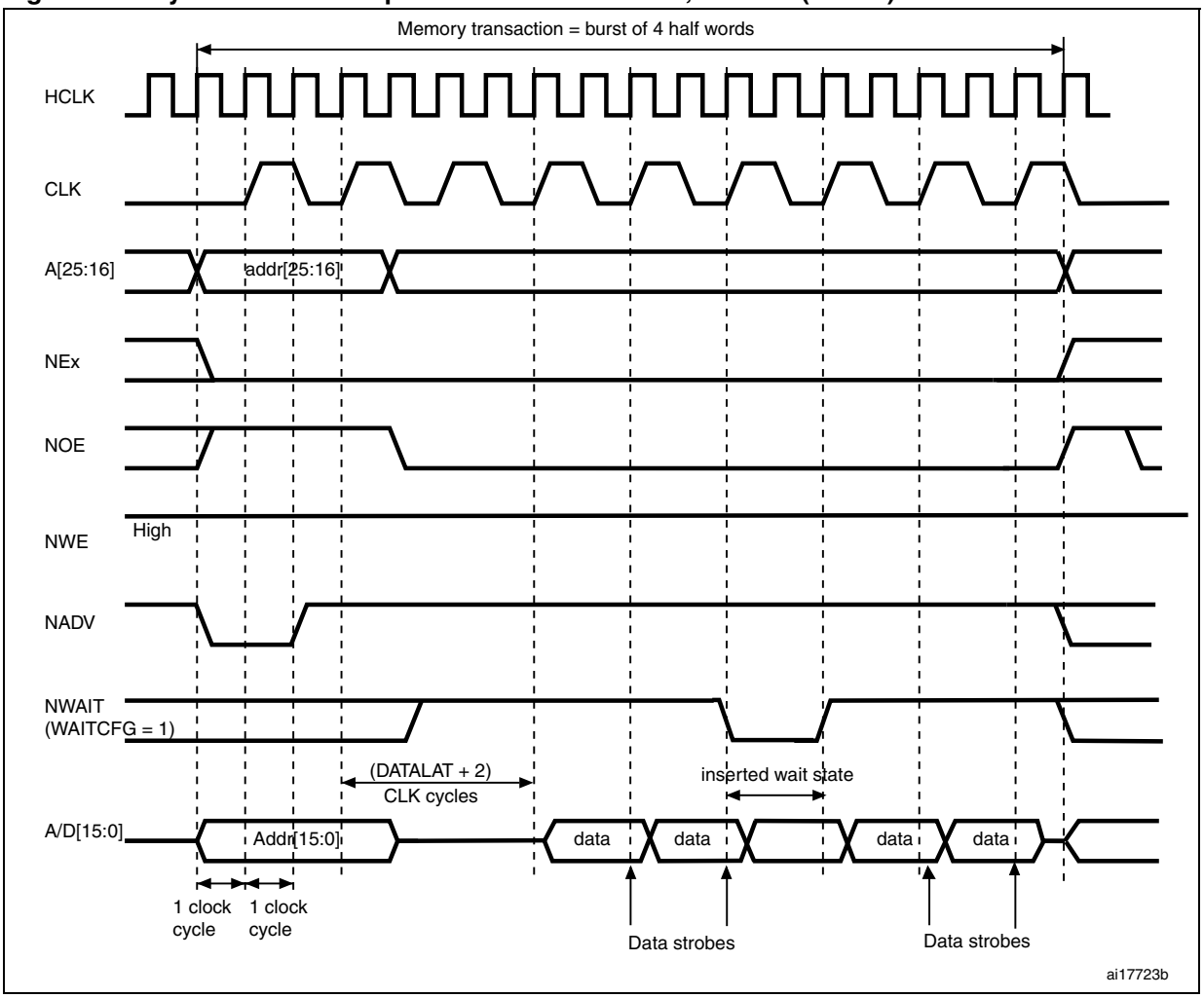


Figure 220. Synchronous multiplexed read mode - NOR, PSRAM (CRAM)



1. Byte lane outputs BL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.

Table 113. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19	CBURSTRW	No effect on synchronous read
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	When high, the first data after latency period is taken as always valid, regardless of the wait from memory value
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory
10	WRAPMOD	no effect
9	WAITPOL	to be set according to memory

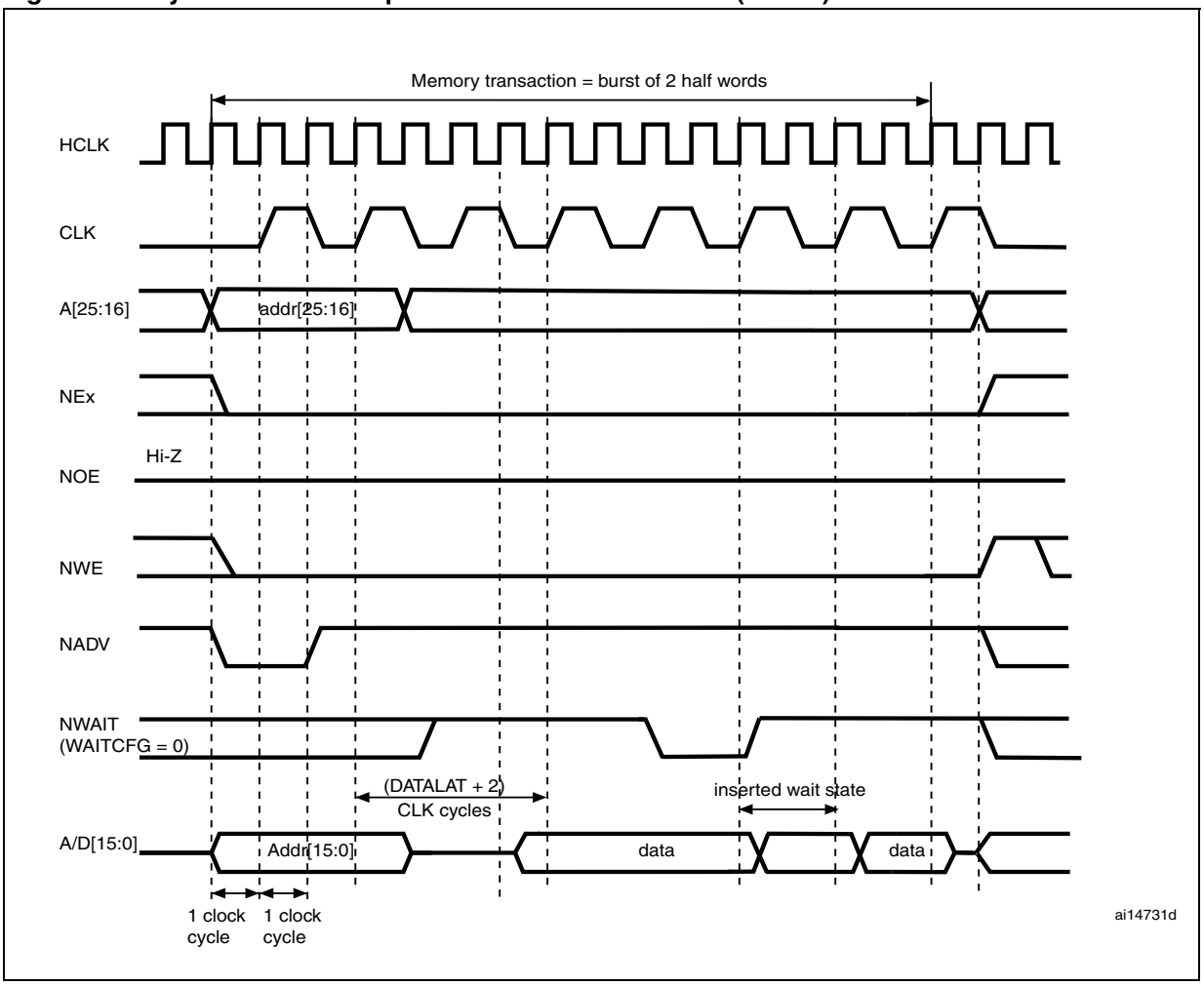
Table 113. FSMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
8	BURSTEN	0x1
7	FWPRLVL	Set to protect memory from accidental write access
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 114. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	no effect
7-4	ADDHLD	no effect
3-0	ADDSET	no effect

Figure 221. Synchronous multiplexed write mode - PSRAM (CRAM)



1. Memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 115. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19	CBURSTRW	0x1
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	When high, the first data after latency period is taken as always valid, regardless of the wait from memory value
12	WREN	no effect on synchronous read
11	WAITCFG	0x0
10	WRAPMOD	no effect
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	FWPRLVL	Set to protect memory from accidental writes
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 116. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30	-	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0 to get CLK = HCLK (not supported) 1 to get CLK = 2 × HCLK
19-16	BUSTURN	No effect
15-8	DATAST	No effect
7-4	ADDHLD	No effect
3-0	ADDSET	No effect

20.5.6 NOR/PSRAM controller registers

SRAM/NOR-Flash chip-select control registers 1..4 (FSMC_BCR1..4)

Address offset: 0xA000 0000 + 8 * (x – 1), x = 1...4

Reset value: 0x0000 30DX

This register contains the control information of each memory bank, used for SRAMs, ROMs and asynchronous or burst NOR Flash memories.

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
Reserved												CBURSTRW	Reserved			ASCYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
												rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 19 **CBURSTRW:** Write burst enable.

For Cellular RAM, the bit enables synchronous burst protocol during write operations. For Flash memory access in burst mode, this bit enables/disables the wait state insertion via the NWAIT signal. The enable bit for the synchronous burst protocol during read access is the BURSTEN bit in the FSMC_BCRx register.

0: Write operations are always performed in asynchronous mode

1: Write operations are performed in synchronous mode.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables the FSMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)

1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 **EXTMOD:** Extended mode enable.

This bit enables the FSMC to program inside the FSMC_BWTR register, so it allows different timings for read and write.

0: values inside FSMC_BWTR register are not taken into account (default after reset)

1: values inside FSMC_BWTR register are taken into account

Bit 13 **WAITEN:** Wait enable bit.

For Flash memory access in burst mode, this bit enables/disables wait-state insertion via the NWAIT signal:

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)

1: NWAIT signal is enabled (its level is taken into account after the programmed Flash latency period to insert wait states if asserted) (default after reset)

Bit 12 **WREN:** Write enable bit.

This bit indicates whether write operations are enabled/disabled in the bank by the FSMC:

0: Write operations are disabled in the bank by the FSMC, an AHB error is reported,

1: Write operations are enabled for the bank by the FSMC (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration.

For memory access in burst mode, the NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:
0: NWAIT signal is active one data cycle before wait state (default after reset),
1: NWAIT signal is active during wait state (not for Cellular RAM).

Bit 10 **WRAPMOD:** Wrapped burst mode support.

Defines whether the controller will or not split an AHB burst wrap access into two linear accesses. Valid only when accessing memories in burst mode
0: Direct wrapped burst is not enabled (default after reset),
1: Direct wrapped burst is enabled.

Note: This bit has no effect as the CPU and DMA cannot generate wrapping burst transfers.

Bit 9 **WAITPOL:** Wait signal polarity bit.

Defines the polarity of the wait signal from memory. Valid only when accessing the memory in burst mode:
0: NWAIT active low (default after reset),
1: NWAIT active high.

Bit 8 **BURSTEN:** Burst enable bit.

Enables the burst access mode for the memory. Valid only with synchronous burst memories:
0: Burst access mode disabled (default after reset)
1: Burst access mode enable

Bit 7 Reserved.

Bit 6 **FACCEN:** Flash access enable

Enables NOR Flash memory access operations.
0: Corresponding NOR Flash memory access is disabled
1: Corresponding NOR Flash memory access is enabled (default after reset)

Bits 5:4 **MWID:** Memory databus width.

Defines the external memory device width, valid for all type of memories.
00: 8 bits,
01: 16 bits (default after reset),
10: reserved, do not use,
11: reserved, do not use.

Bits 3:2 **MTYP:** Memory type.

Defines the type of external memory attached to the corresponding memory bank:
00: SRAM, ROM (default after reset for Bank 2...4)
01: PSRAM (Cellular RAM: CRAM)
10: NOR Flash/OneNAND Flash (default after reset for Bank 1)
11: reserved

Bit 1 **MUXEN:** Address/data multiplexing enable bit.

When this bit is set, the address and data values are multiplexed on the databus, valid only with NOR and PSRAM memories:
0: Address/Data nonmultiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory bank enable bit.

Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.
0: Corresponding memory bank is disabled
1: Corresponding memory bank is enabled

SRAM/NOR-Flash chip-select timing registers 1..4 (FSMC_BTR1..4)

Address offset: 0xA000 0000 + 0x04 + 8 * (x – 1), x = 1..4

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, ROMs and NOR Flash memories. If the EXTMOD bit is set in the FSMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FSMC_BWTRx registers).

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
Reserved		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
		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

Bits 29:28 **ACCMOD:** Access mode

Specifies the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FSMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:24 **DATLAT**: Data latency for synchronous burst NOR Flash memory

For NOR Flash with synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:

0000: Data latency of 2 CLK clock cycles for first burst access

1111: Data latency of 17 CLK clock cycles for first burst access (default value after reset)

Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (CLK) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In the case of CRAM, this field must be set to '0'.

Bits 23:20 **CLKDIV:** Clock divide ratio (for CLK signal)

Defines the period of CLK clock output signal, expressed in number of HCLK cycles:

0000: Reserved

0001: CLK period = 2 × HCLK periods

0010: CLK period = 3 × HCLK periods

1111: CLK period = 16 × HCLK periods (default value after reset)

In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care.

Bits 19:16 **BUSTURN:** Bus turnaround phase duration

These bits are written by software to insert the bus turnaround delay after a read access only from multiplexed NOR Flash memory to avoid bus contention if the controller needs to drive addresses on the databus for the next side-by-side transaction. BUSTURN can be set to the minimum if the slowest memory does not take more than 6 HCLK clock cycles to put the databus in Hi-Z state.

These bits are written by software to add a delay at the end of a write/read transaction. This delay allows to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (tEHQZ):

(BUSTRUN + 1)HCLK period ≥ $t_{EHELmin}$ and (BUSTRUN + 2)HCLK period ≥ $t_{EHQZmax}$ if EXTMOD = '0'

(BUSTRUN + 2)HCLK period ≥ max ($t_{EHELmin}$, $t_{EHQZmax}$) if EXTMOD = '1'.

0000: BUSTURN phase duration = 0 HCLK clock cycle added

...

1111: BUSTURN phase duration = 15 × HCLK clock cycles (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 204* to *Figure 216*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × HCLK clock cycles

0000 0010: DATAST phase duration = 2 × HCLK clock cycles

...

1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

For each memory type and access mode data-phase duration, please refer to the respective figure (*Figure 204* to *Figure 216*).

Example: Mode1, write access, DATAST=1: Data-phase duration= DATAST+1 = 2 HCLK clock cycles.

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 213* to *Figure 216*), used in mode D and multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration =1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

For each access mode address-hold phase duration, please refer to the respective figure (*Figure 213* to *Figure 216*).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 **ADDSET:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 204* to *Figure 216*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

1111: ADDSET phase duration = 1615 × HCLK clock cycles (default value after reset)

For each access mode address setup phase duration, please refer to the respective figure (refer to *Figure 204* to *Figure 216*).

Note: In synchronous accesses, this value is don't care.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.
With PSRAMs (CRAMs) the filed DATLAT must be set to 0, so that the FSMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).

SRAM/NOR-Flash write timing registers 1..4 (FSMC_BWTR1..4)

Address offset: 0xA000 0000 + 0x104 + 8 * (x – 1), x = 1...4

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, ROMs and NOR Flash memories. When the EXTMOD bit is set in the FSMC_BCRx register, then this register is active for write access.

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
Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
		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

Bits 29:28 **ACCMOD:** Access mode.
Specifies the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FSMC_BCRx register is 1.
00: access mode A
01: access mode B
10: access mode C
11: access mode D

Bits 27:24 **DATLAT:** Data latency (for synchronous burst NOR Flash).
For NOR Flash with Synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:
0000: (0x0) Data latency of 2 CLK clock cycles for first burst access
...
1111: (0xF) Data latency of 17 CLK clock cycles for first burst access (default value after reset)
*Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (**CLK**) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In case of CRAM, this field must be set to 0*

Bits 23:20 **CLKDIV:** Clock divide ratio (for CLK signal).
Defines the period of CLK clock output signal, expressed in number of HCLK cycles:
0000: Reserved
0001 CLK period = 2 × HCLK periods
0010 CLK period = 3 × HCLK periods
1111: CLK period = 16 × HCLK periods (default value after reset)
In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care.

Bits 19:16 **BUSTURN**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (t_{EHEL} from ENx high to ENx low):
(BUSTRUN + 1) HCLK period ≥ $t_{EHELmin}$.
0000: BUSTURN phase duration = 0 HCLK clock cycle added
...
1111: BUSTURN phase duration = 15 HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to *Figure 204* to *Figure 216*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:
0000 0000: Reserved
0000 0001: DATAST phase duration = 1 × HCLK clock cycles
0000 0010: DATAST phase duration = 2 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD:** Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 213* to *Figure 216*), used in SRAMs, ROMs and asynchronous multiplexed NOR Flash accesses:
0000: Reserved
0001: ADDHLD phase duration = 1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
...
1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is not used, the address hold phase is always 1 Flash clock period duration.

Bits 3:0 **ADDSET:** Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to *Figure 213* to *Figure 216*), used in SRAMs, ROMs and asynchronous NOR Flash accessed:
0000: ADDSET phase duration = 0 × HCLK clock cycle
...
1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is don't care.

20.5.7 FSMC register map

The following table summarizes the FSMC registers.

Table 117. FSMC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA000 0000	FSMC_BCR1	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG		WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
	Reset value																																
0xA000 0008	FSMC_BCR2	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0010	FSMC_BCR3	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0018	FSMC_BCR4	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0004	FSMC_BTR1	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 000C	FSMC_BTR2	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0014	FSMC_BTR3	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 001C	FSMC_BTR4	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0104	FSMC_BWTR1	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
0xA000 010C	FSMC_BWTR2	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
0xA000 0114	FSMC_BWTR3	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
0xA000 011C	FSMC_BWTR4	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			

Refer to for the register boundary addresses.

21 Serial peripheral interface (SPI)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

21.1 SPI introduction

The serial peripheral interface (SPI) allows half/ full-duplex, synchronous, serial communication with external devices. The interface can be configured as the 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.

It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking.

Warning: **Since some SPI1and SPI3 pins may be mapped onto some pins used by the JTAG interface (SPI1/3_NSS onto JTDI, SPI1/3_SCK onto JTDO and SPI1/3_MISO onto NJTRST), you may either:**
– disable the JTAG and use the SWD interface prior to configuring the pins listed as SPI IOs (when debugging the application) or
– disable both JTAG/SWD interfaces (for standalone applications).
For more information on the configuration of the JTAG/SWD interface pins, please refer to *Section 7.3.3: JTAG/SWD alternate function remapping*.

21.2 SPI main features

21.2.1 SPI features

- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines with or without a bidirectional data line
- 8- or 16-bit transfer frame format selection
- Master or slave operation
- Multimaster mode capability
- 8 master mode baud rate prescalers (f_{PCLK}/2 max.)
- Slave mode frequency (f_{PCLK}/2 max)
- Faster communication for both master and slave
- 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
- 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 and CRC error flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability: Tx and Rx requests

21.3 SPI functional description

21.3.1 General description

The block diagram of the SPI is shown in *Figure 222*.

Figure 222. SPI block diagram

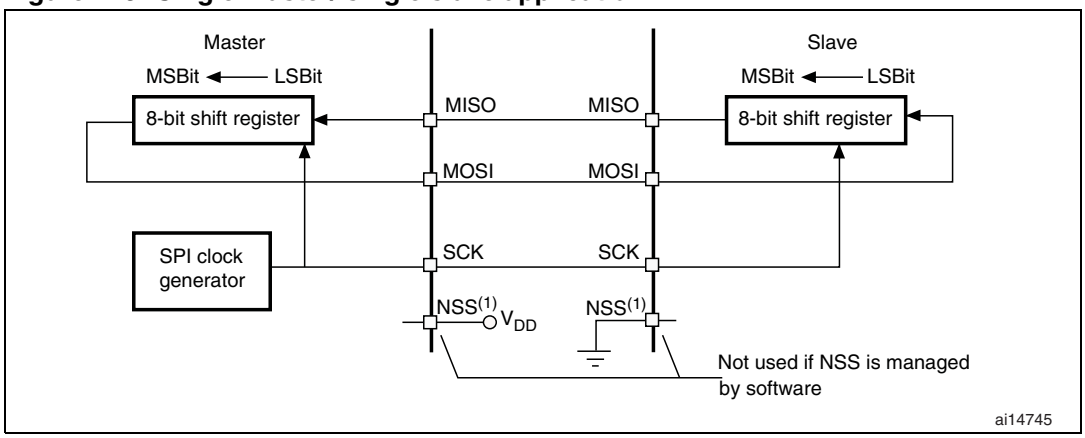
Usually, the SPI is connected to external devices through 4 pins:

- MISO: Master In / Slave Out data. This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out / Slave In data. This pin can be used to transmit data in master mode and receive data in slave mode.
- SCK: Serial Clock output for SPI masters and input for SPI slaves.
- NSS: Slave select. This is an optional pin to select a slave device. This pin acts as a ‘chip select’ to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard IO ports on the master device. The NSS pin may also be used as an output if enabled (SSOE bit) and driven low if the SPI is in master configuration. In this manner, all NSS pins from devices connected to the Master NSS pin see a low level and become slaves when they are configured in NSS hardware mode. When configured in master mode with NSS configured as an input (MSTR=1 and SSOE=0) and if NSS is pulled low, the SPI

enters the master mode fault state: the MSTR bit is automatically cleared and the device is configured in slave mode (refer to *Section 21.3.10: Error flags on page 533*).

A basic example of interconnections between a single master and a single slave is illustrated in *Figure 223*.

Figure 223. Single master/ single slave application



1. Here, the NSS pin is configured as an input.

The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin. This implies full-duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI_CR1 register.

- Software NSS management (SSM = 1)

  The slave select information is driven internally by the value of the SSI bit in the SPI_CR1 register. The external NSS pin remains free for other application uses.

- Hardware NSS management (SSM = 0)

  Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI_CR1).

  – NSS output enabled (SSM = 0, SSOE = 1)

    This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.

  – NSS output disabled (SSM = 0, SSOE = 0)

    This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS high.

Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CR1 register. The CPOL (clock polarity) bit controls the steady 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 (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data are latched on the occurrence of the second clock transition. If the CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data are latched on the occurrence of the first clock transition.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 224, shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

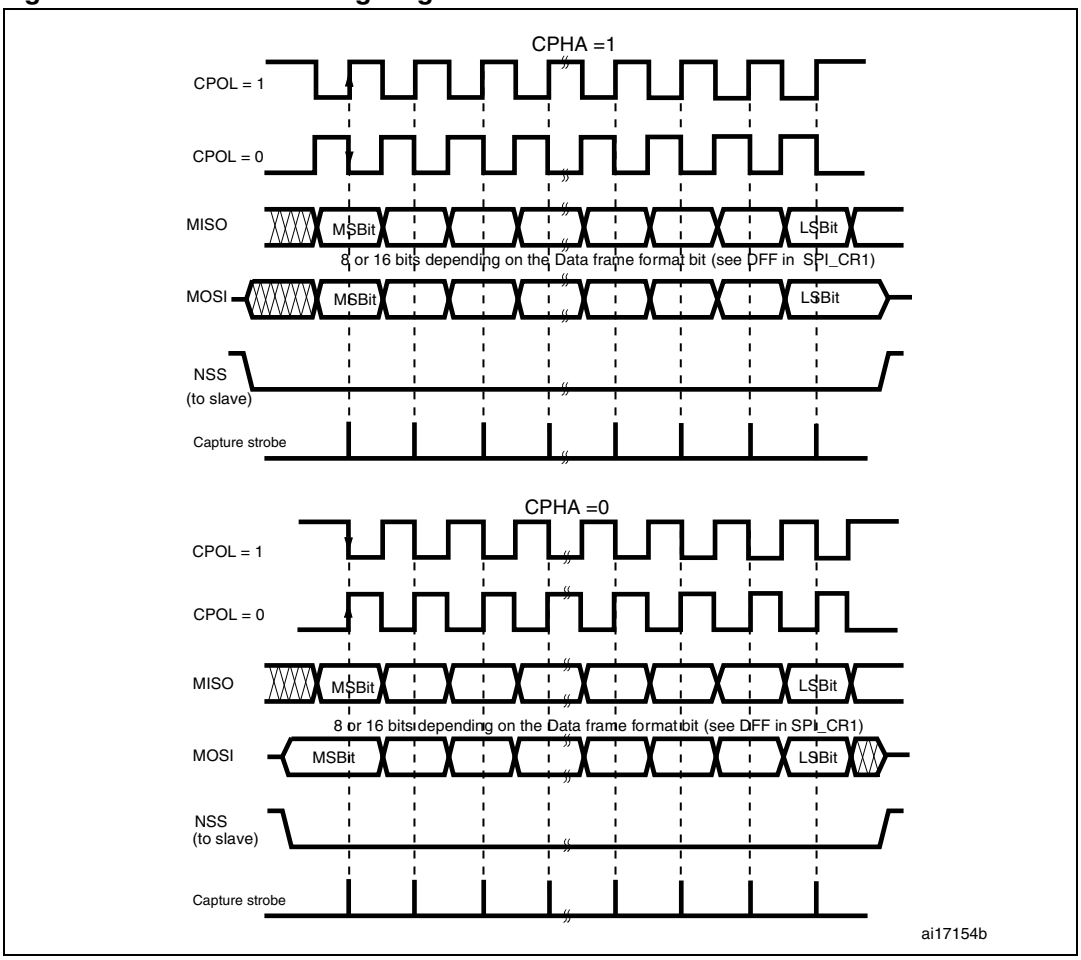
Note: 1 Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

2 Master and slave must be programmed with the same timing mode.

3 The idle state of SCK must correspond to the polarity selected in the SPI_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

4 The Data Frame Format (8- or 16-bit) is selected through the DFF bit in SPI_CR1 register, and determines the data length during transmission/reception.

Figure 224. Data clock timing diagram



1. These timings are shown with the LSBFIRST bit reset in the SPI_CR1 register.

Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 Register.

Each data frame is 8 or 16 bits long depending on the size of the data programmed using the DFF bit in the SPI_CR1 register. The selected data frame format is applicable for transmission and/or reception.

21.3.2 Configuring the SPI in slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the BR[2:0] bits in the SPI_CR1 register, does not affect the data transfer rate.

Note: 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 needs to be ready before the first edge of the communication clock or before the end of the ongoing communication. It is mandatory to have the polarity of the communication clock set to the steady state value before the slave and the master are enabled.

Follow the procedure below to configure the SPI in slave mode:

Procedure

1. Set the DFF bit to define 8- or 16-bit data frame format
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see *Figure 224*). For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device.
3. The frame format (MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 register) must be the same as the master device.
4. In Hardware mode (refer to *Slave select (NSS) pin management on page 516*), the NSS pin must be connected to a low level signal during the complete byte transmit sequence. In NSS software mode, set the SSM bit and clear the SSI bit in the SPI_CR1 register.
5. Clear the MSTR bit and set the SPE bit (both in the SPI_CR1 register) to assign the pins to alternate functions.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Transmit sequence

The data byte is parallel-loaded into the Tx buffer during a write cycle.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin. The remaining bits (the 7 bits in 8-bit data frame format, and the 15 bits in 16-bit data frame format) are loaded into the shift-register. The TXE flag in the SPI_SR register is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The Data in shift register is transferred to Rx Buffer and the RXNE flag (SPI_SR register) is set
- An Interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register.

After the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing of the RXNE bit is performed by reading the SPI_DR register.

21.3.3 Configuring the SPI in master mode

In the master configuration, the serial clock is generated on the SCK pin.

Procedure

1. Select the BR[2:0] bits to define the serial clock baud rate (see SPI_CR1 register).
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see *Figure 224*).
3. Set the DFF bit to define 8- or 16-bit data frame format
4. Configure the LSBFIRST bit in the SPI_CR1 register to define the frame format.
5. If the NSS pin is required in input mode, in hardware mode, connect the NSS pin to a high-level signal during the complete byte transmit sequence. In NSS software mode, set the SSM and SSI bits in the SPI_CR1 register. If the NSS pin is required in output mode, the SSOE bit only should be set.
6. The MSTR and SPE bits must be set (they remain set only if the NSS pin is connected to a high-level signal).

In this configuration the MOSI pin is a data output and the MISO pin is a data input.

Transmit sequence

The transmit sequence begins when a byte is written in the Tx Buffer.

The data byte is parallel-loaded into the shift register (from the internal bus) during the first bit transmission and then shifted out serially to the MOSI pin MSB first or LSB first depending on the LSBFIRST bit in the SPI_CR1 register. The TXE flag is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The data in the shift register is transferred to the RX Buffer and the RXNE flag is set
- An interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register

At the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing the RXNE bit is performed by reading the SPI_DR register.

A continuous transmit stream can be maintained if the next data to be transmitted is put in the Tx buffer once the transmission is started. Note that TXE flag should be ‘1 before any attempt to write the Tx buffer is made.

Note: When a master is communicating with SPI slaves which need to be de-selected between transmissions, the NSS pin must be configured as GPIO or another GPIO must be used and toggled by software.

21.3.4 Configuring the SPI for simplex communication

The SPI is capable of operating in simplex mode in 2 configurations.

- 1 clock and 1 bidirectional data wire
- 1 clock and 1 data wire (receive-only or transmit-only)

1 clock and 1 bidirectional data wire (BIDIMODE=1)

This mode is enabled by setting the BIDIMODE bit in the SPI_CR1 register. In this mode SCK is used for the clock and MOSI in master or MISO in slave mode is used for data communication. The transfer direction (Input/Output) is selected by the BIDIOE bit in the SPI_CR1 register. When this bit is 1, the data line is output otherwise it is input.

1 clock and 1 unidirectional data wire (BIDIMODE=0)

In this mode, the application can use the SPI either in transmit-only mode or in receive-only mode.

- Transmit-only mode is similar to full-duplex mode (BIDIMODE=0, RXONLY=0): the data are transmitted on the transmit pin (MOSI in master mode or MISO in slave mode) and the receive pin (MISO in master mode or MOSI in slave mode) can be used as a general-purpose IO. In this case, the application just needs to ignore the Rx buffer (if the data register is read, it does not contain the received value).
- In receive-only mode, the application can disable the SPI output function by setting the RXONLY bit in the SPI_CR2 register. In this case, it frees the transmit IO pin (MOSI in master mode or MISO in slave mode), so it can be used for other purposes.

To start the communication in receive-only mode, configure and enable the SPI:

- In master mode, the communication starts immediately and stops when the SPE bit is cleared and the current reception stops. There is no need to read the BSY flag in this mode. It is always set when an SPI communication is ongoing.
- In slave mode, the SPI continues to receive as long as the NSS is pulled down (or the SSI bit is cleared in NSS software mode) and the SCK is running.

21.3.5 Data transmission and reception procedures

Rx and Tx buffers

In reception, data are received and then stored into an internal Rx buffer while In transmission, data are first stored into an internal Tx buffer before being transmitted.

A read access of the SPI_DR register returns the Rx buffered value whereas a write access to the SPI_DR stores the written data into the Tx buffer.

Start sequence in master mode

- In full-duplex (BIDIMODE=0 and RXONLY=0)
  - The sequence begins when data are written into the SPI_DR register (Tx buffer).
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - At the same time, the received data on the MISO pin is shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - The sequence begins as soon as SPE=1
  - Only the receiver is activated and the received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
  - The sequence begins when data are written into the SPI_DR register (Tx buffer).
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - No data are received.
- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
  - The sequence begins as soon as SPE=1 and BIDIOE=0.
  - The received data on the MOSI pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
  - The transmitter is not activated and no data are shifted out serially to the MOSI pin.

Start sequence in slave mode

- In full-duplex mode (BIDIMODE=0 and RXONLY=0)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
  - At the same time, the data are parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
  - The transmitter is not activated and no data are shifted out serially to the MISO pin.
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
  - The sequence begins when the slave device receives the clock signal and the first bit in the Tx buffer is transmitted on the MISO pin.
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
  - No data are received.

- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MISO pin.
  - The received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
  - The transmitter is not activated and no data are shifted out serially to the MISO pin.

Handling data transmission and reception

The TXE flag (Tx buffer empty) is set when the data are transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data. An interrupt can be generated if the TXEIE bit in the SPI_CR2 register is set. Clearing the TXE bit is performed by writing to the SPI_DR register.

Note: The software must ensure that the TXE flag is set to 1 before attempting to write to the Tx buffer. Otherwise, it overwrites the data previously written to the Tx buffer.

The RXNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data are transferred from the shift register to the Rx buffer. It indicates that data are ready to be read from the SPI_DR register. An interrupt can be generated if the RXNEIE bit in the SPI_CR2 register is set. Clearing the RXNE bit is performed by reading the SPI_DR register.

For some configurations, the BSY flag can be used during the last data transfer to wait until the completion of the transfer.

Full-duplex transmit and receive procedure in master or slave mode (BIDIMODE=0 and RXONLY=0)

The software has to follow this procedure to transmit and receive data (see *Figure 225* and *Figure 226*):

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to be transmitted into the SPI_DR register (this clears the TXE flag).
3. Wait until TXE=1 and write the second data item to be transmitted. Then wait until RXNE=1 and read the SPI_DR to get the first received data item (this clears the RXNE bit). Repeat this operation for each data item to be transmitted/received until the n–1 received data.
4. Wait until RXNE=1 and read the last received data.
5. Wait until TXE=1 and then wait until BSY=0 before disabling the SPI.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edges of the RXNE or TXE flag.

Figure 225. TXE/RXNE/BSY behavior in Master / full-duplex mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers

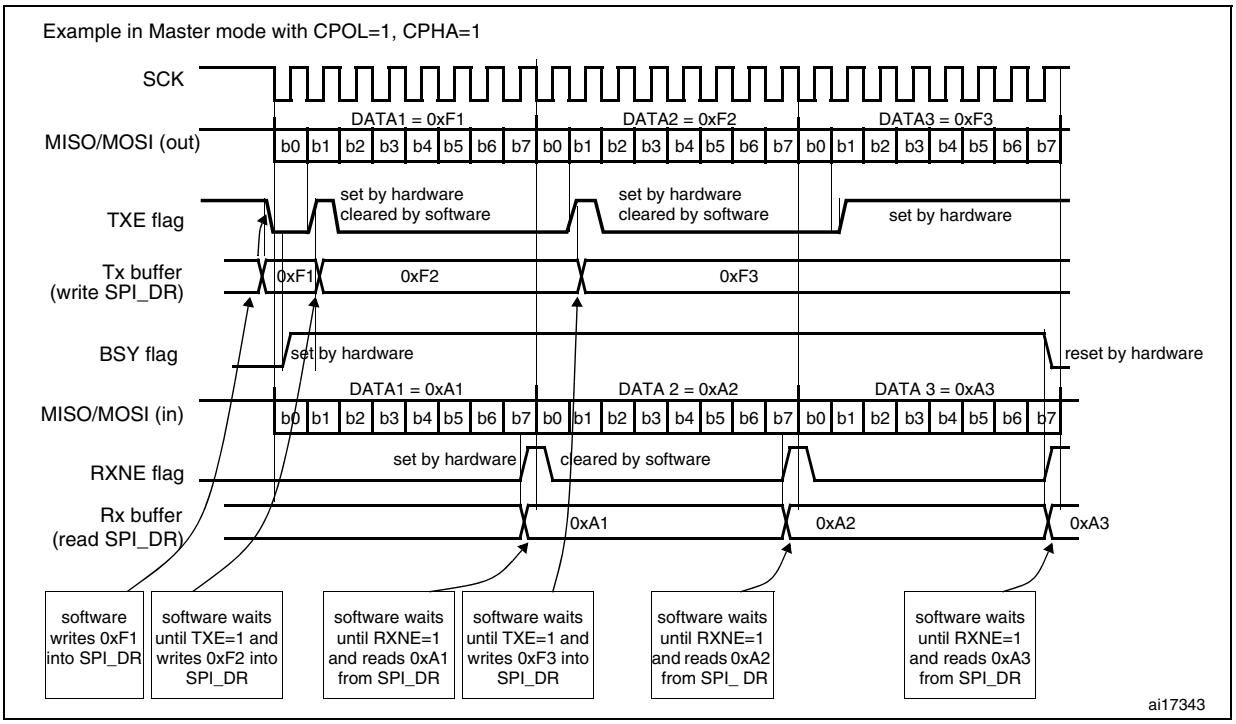
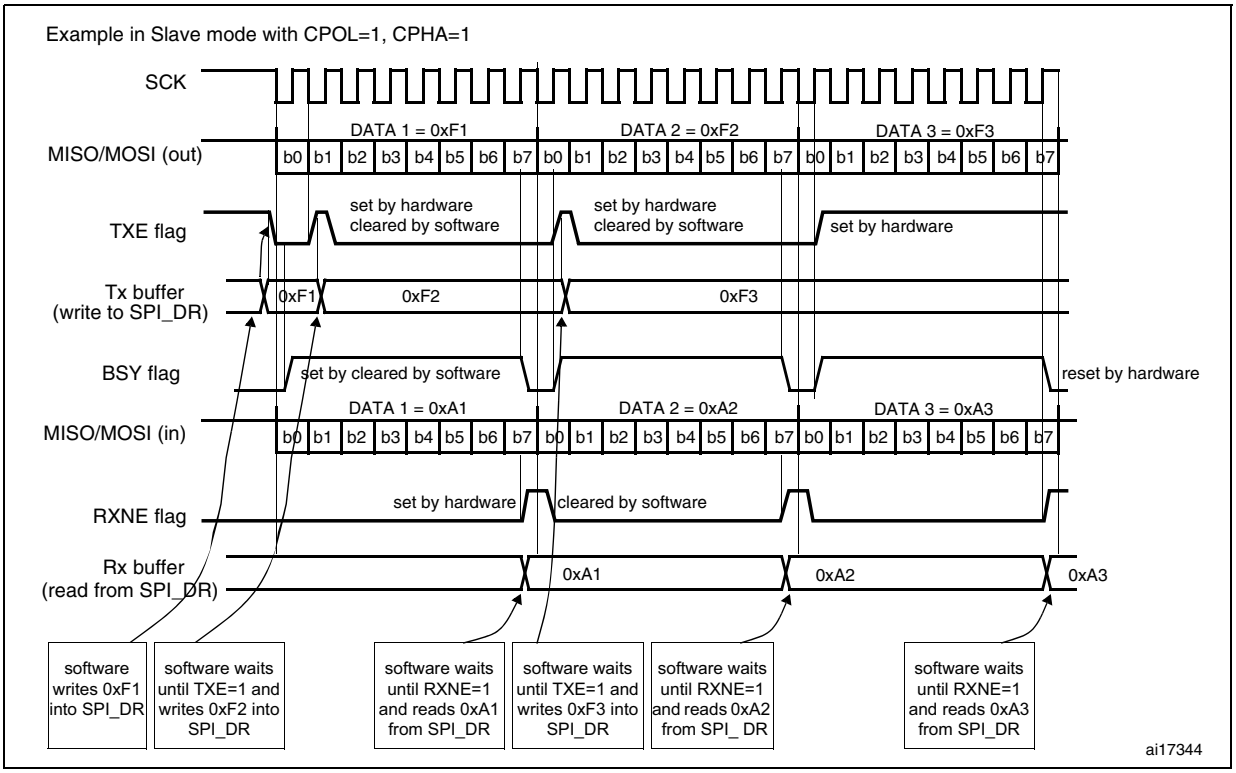


Figure 226. TXE/RXNE/BSY behavior in Slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in the case of continuous transfers



Transmit-only procedure (BIDIMODE=0 RXONLY=0)

In this mode, the procedure can be reduced as described below and the BSY bit can be used to wait until the completion of the transmission (see *Figure 227* and *Figure 228*).

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to send into the SPI_DR register (this clears the TXE bit).
3. Wait until TXE=1 and write the next data item to be transmitted. Repeat this step for each data item to be transmitted.
4. After writing the last data item into the SPI_DR register, wait until TXE=1, then wait until BSY=0, this indicates that the transmission of the last data is complete.

This procedure can be also implemented using dedicated interrupt subroutines launched at each rising edge of the TXE flag.

Note: 1 During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, in transmit-only mode, it is mandatory to wait first until TXE is set and then until BSY is cleared after writing the last data.

2 After transmitting two data items in transmit-only mode, the OVR flag is set in the SPI_SR register since the received data are never read.

Figure 227. TXE/BSY behavior in Master transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers

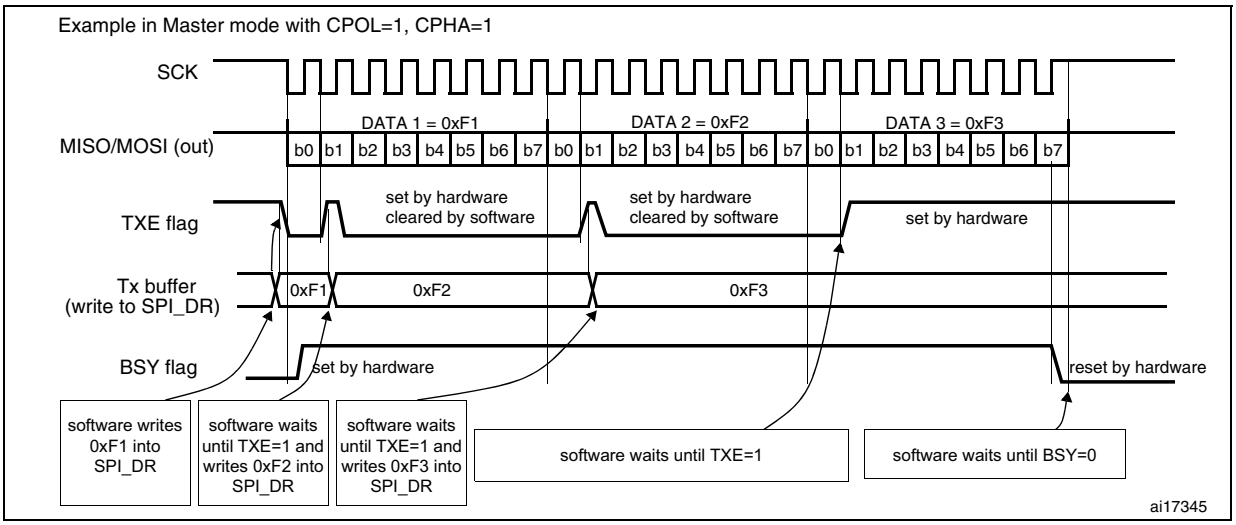
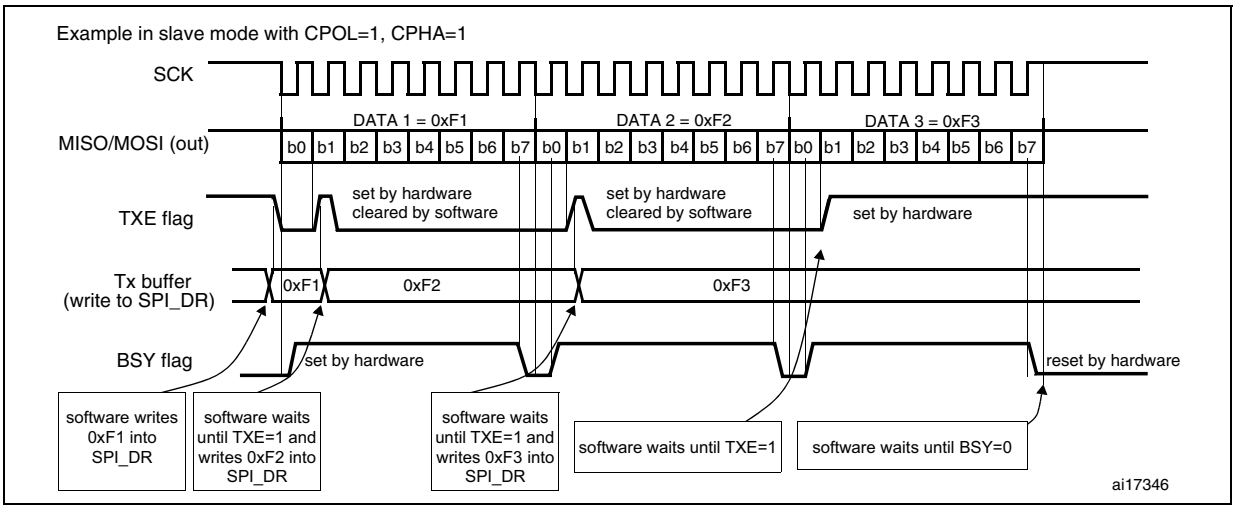


Figure 228. TXE/BSY in Slave transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers



Bidirectional transmit procedure (BIDIMODE=1 and BIDIOE=1)

In this mode, the procedure is similar to the procedure in Transmit-only mode except that the BIDIMODE and BIDIOE bits both have to be set in the SPI_CR2 register before enabling the SPI.

Unidirectional receive-only procedure (BIDIMODE=0 and RXONLY=1)

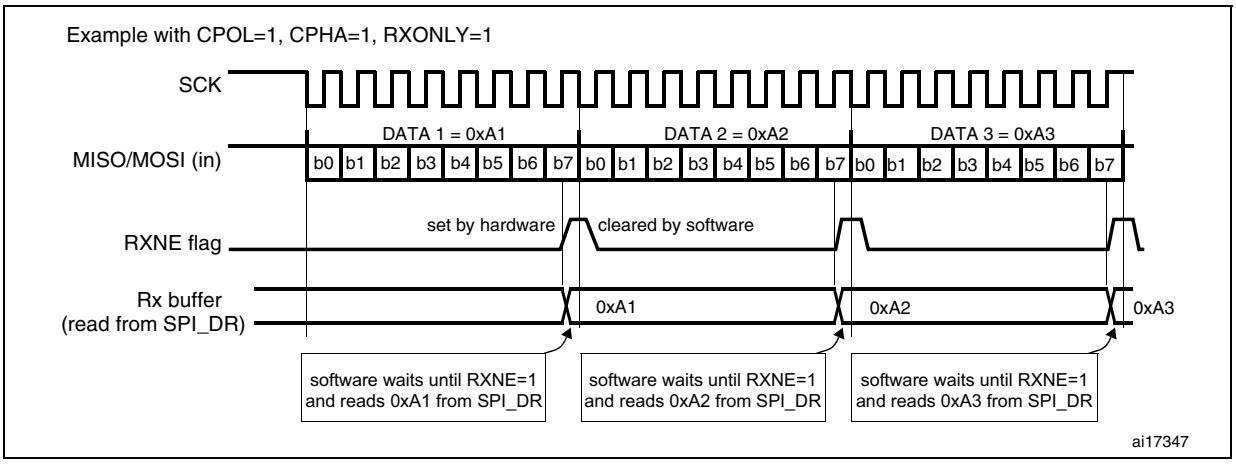
In this mode, the procedure can be reduced as described below (see *Figure 229*):

1. Set the RXONLY bit in the SPI_CR2 register.
2. Enable the SPI by setting the SPE bit to 1:
   a) In master mode, this immediately activates the generation of the SCK clock, and data are serially received until the SPI is disabled (SPE=0).
   b) In slave mode, data are received when the SPI master device drives NSS low and generates the SCK clock.
3. Wait until RXNE=1 and read the SPI_DR register to get the received data (this clears the RXNE bit). Repeat this operation for each data item to be received.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edge of the RXNE flag.

Note: If it is required to disable the SPI after the last transfer, follow the recommendation described in Section 21.3.8: Disabling the SPI on page 531.

Figure 229. RXNE behavior in receive-only mode (BIDIRMODE=0 and RXONLY=1) in the case of continuous transfers



Bidirectional receive procedure (BIDIMODE=1 and BIDIOE=0)

In this mode, the procedure is similar to the Receive-only mode procedure except that the BIDIMODE bit has to be set and the BIDIOE bit cleared in the SPI_CR2 register before enabling the SPI.

Continuous and discontinuous transfers

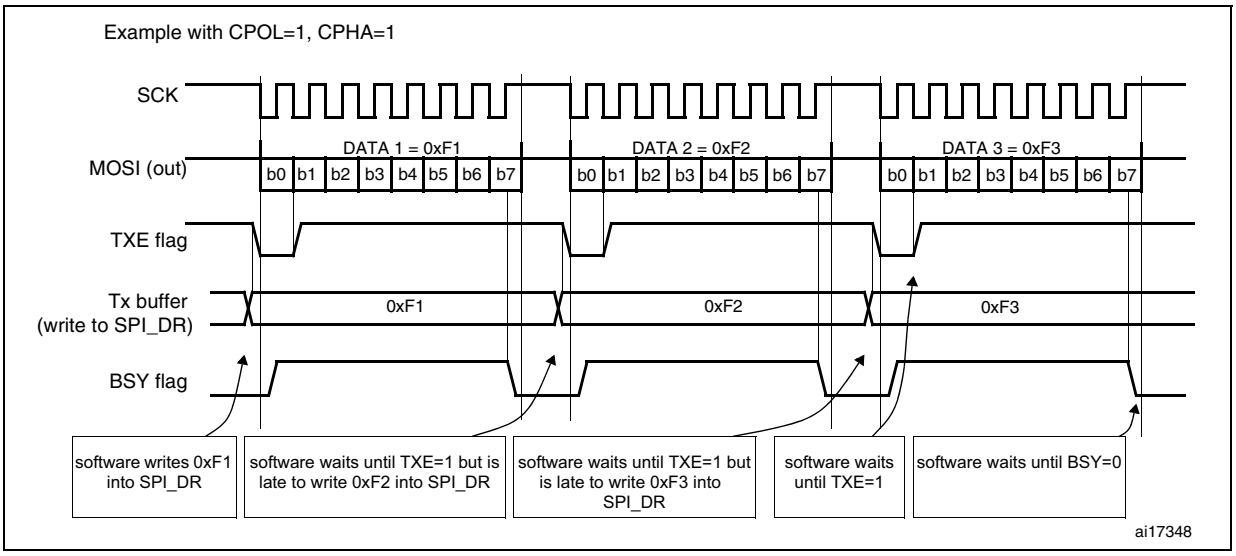
When transmitting data in master mode, if the software is fast enough to detect each rising edge of TXE (or TXE interrupt) and to immediately write to the SPI_DR register before the ongoing data transfer is complete, the communication is said to be continuous. In this case, there is no discontinuity in the generation of the SPI clock between each data item and the BSY bit is never cleared between each data transfer.

On the contrary, if the software is not fast enough, this can lead to some discontinuities in the communication. In this case, the BSY bit is cleared between each data transmission (see *Figure 230*).

In Master receive-only mode (RXONLY=1), the communication is always continuous and the BSY flag is always read at 1.

In slave mode, the continuity of the communication is decided by the SPI master device. In any case, even if the communication is continuous, the BSY flag goes low between each transfer for a minimum duration of one SPI clock cycle (see *Figure 228*).

Figure 230. TXE/BSY behavior when transmitting (BIDIRMODE=0 and RXONLY=0) in the case of discontinuous transfers



21.3.6 CRC calculation

A CRC calculator has been implemented for communication reliability. Separate CRC calculators are implemented for transmitted data and received data. The CRC is calculated using a programmable polynomial serially on each bit. It is calculated on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register.

Note: This SPI offers two kinds of CRC calculation standard which depend directly on the data frame format selected for the transmission and/or reception: 8-bit data (CR8) and 16-bit data (CRC16).

CRC calculation is enabled by setting the CRCEN bit in the SPI_CR1 register. This action resets the CRC registers (SPI_RXCRCR and SPI_TXCRCR). In full duplex or transmitter only mode, when the transfers are managed by the software (CPU mode), it is necessary to write the bit CRCNEXT immediately after the last data to be transferred is written to the SPI_DR. At the end of this last data transfer, the SPI_TXCRCR value is transmitted.

In receive only mode and when the transfers are managed by software (CPU mode), it is necessary to write the CRCNEXT bit after the second last data has been received. The CRC is received just after the last data reception and the CRC check is then performed.

At the end of data and CRC transfers, the CRCERR flag in the SPI_SR register is set if corruption occurs during the transfer.

If data are present in the TX buffer, the CRC value is transmitted only after the transmission of the data byte. During CRC transmission, the CRC calculator is switched off and the register value remains unchanged.

Note: Please refer to the product datasheet for availability of this feature.

SPI communication using the CRC is possible through the following procedure:

1. Program the CPOL, CPHA, LSBFirst, BR, SSM, SSI and MSTR values.
2. Program the polynomial in the SPI_CRCPR register.
3. Enable the CRC calculation by setting the CRCEN bit in the SPI_CR1 register. This also clears the SPI_RXCRCR and SPI_TXCRCR registers.
4. Enable the SPI by setting the SPE bit in the SPI_CR1 register.
5. Start the communication and sustain the communication until all but one byte or half-word have been transmitted or received.
   - In full duplex or transmitter-only mode, when the transfers are managed by software, when writing the last byte or half word to the Tx buffer, set the CRCNEXT bit in the SPI_CR1 register to indicate that the CRC will be transmitted after the transmission of the last byte.
   - In receiver only mode, set the bit CRCNEXT just after the reception of the second to last data to prepare the SPI to enter in CRC Phase at the end of the reception of the last data. CRC calculation is frozen during the CRC transfer.
6. After the transfer of the last byte or half word, the SPI enters the CRC transfer and check phase. In full duplex mode or receiver-only mode, the received CRC is compared to the SPI_RXCRCR value. If the value does not match, the CRCERR flag in SPI_SR is set and an interrupt can be generated when the ERRIE bit in the SPI_CR2 register is set.

Note: When the SPI is in slave mode, be careful to enable CRC calculation only when the clock is stable, that is, when the clock is in the steady state. If not, a wrong CRC calculation may be done. In fact, the CRC is sensitive to the SCK slave input clock as soon as CRCEN is set, and this, whatever the value of the SPE bit.

With high bitrate frequencies, be careful when transmitting the CRC. As the number of used CPU cycles has to be as low as possible in the CRC transfer phase, it is forbidden to call software functions in the CRC transmission sequence to avoid errors in the last data and CRC reception. In fact, CRCNEXT bit has to be written before the end of the transmission/reception of the last data.

For high bit rate frequencies, it is advised to use the DMA mode to avoid the degradation of the SPI speed performance due to CPU accesses impacting the SPI bandwidth.

When the STM32F100xx are configured as slaves and the NSS hardware mode is used, the NSS pin needs to be kept low between the data phase and the CRC phase.

When the SPI is configured in slave mode with the CRC feature enabled, CRC calculation takes place even if a high level is applied on the NSS pin. This may happen for example in case of a multislave environment where the communication master addresses slaves alternately.

Between a slave deselection (high level on NSS) and a new slave selection (low level on NSS), the CRC value should be cleared on both master and slave sides in order to resynchronize the master and slave for their respective CRC calculation.

To clear the CRC, follow the procedure below:

1. Disable SPI (SPE = 0)
2. Clear the CRCEN bit
3. Set the CRCEN bit
4. Enable the SPI (SPE = 1)

21.3.7 Status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

When it is set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can be loaded into the buffer. The TXE flag is cleared when writing to the SPI_DR register.

Rx buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the Rx buffer. It is cleared when SPI_DR is read.

BUSY flag

This BSY flag is set and cleared by hardware (writing to this flag has no effect). The BSY flag indicates the state of the communication layer of the SPI.

When BSY is set, it indicates that the SPI is busy communicating. There is one exception in master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software wants to disable the SPI and enter Halt mode (or disable the peripheral clock). This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is also useful to avoid write collisions in a multimaster system.

The BSY flag is set when a transfer starts, with the exception of master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0).

It is cleared:

- when a transfer is finished (except in master mode if the communication is continuous)
- when the SPI is disabled
- when a master mode fault occurs (MODF=1)

When communication is not continuous, the BSY flag is low between each communication.

When communication is continuous:

- in master mode, the BSY flag is kept high during all the transfers
- in slave mode, the BSY flag goes low for one SPI clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

21.3.8 Disabling the SPI

When a transfer is terminated, the application can stop the communication by disabling the SPI peripheral. This is done by clearing the SPE bit.

For some configurations, disabling the SPI and entering the Halt mode while a transfer is ongoing can cause the current transfer to be corrupted and/or the BSY flag might become unreliable.

To avoid any of those effects, it is recommended to respect the following procedure when disabling the SPI:

In master or slave full-duplex mode (BIDIMODE=0, RXONLY=0)

1. Wait until RXNE=1 to receive the last data
2. Wait until TXE=1
3. Then wait until BSY=0
4. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master or slave unidirectional transmit-only mode (BIDIMODE=0, RXONLY=0) or bidirectional transmit mode (BIDIMODE=1, BIDIOE=1)

After the last data is written into the SPI_DR register:

1. Wait until TXE=1
2. Then wait until BSY=0
3. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master unidirectional receive-only mode (MSTR=1, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=1, BIDIMODE=1, BIDIOE=0)

This case must be managed in a particular way to ensure that the SPI does not initiate a new transfer:

1. Wait for the second to last occurrence of RXNE=1 (n–1)
2. Then wait for one SPI clock cycle (using a software loop) before disabling the SPI (SPE=0)
3. Then wait for the last RXNE=1 before entering the Halt mode (or disabling the peripheral clock)

Note: In master bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0), the BSY flag is kept low during transfers.

In slave receive-only mode (MSTR=0, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=0, BIDIMODE=1, BIDOE=0)

1. You can disable the SPI (write SPE=1) at any time: the current transfer will complete before the SPI is effectively disabled
2. Then, if you want to enter the Halt mode, you must first wait until BSY = 0 before entering the Halt mode (or disabling the peripheral clock).

21.3.9 SPI communication using DMA (direct memory addressing)

To operate at its maximum speed, the SPI needs to be fed with the data for transmission and the data received on the Rx buffer should be read to avoid overrun. To facilitate the transfers, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

A DMA access is requested when the enable bit in the SPI_CR2 register is enabled. Separate requests must be issued to the Tx and Rx buffers (see *Figure 231* and *Figure 232*):

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPI_DR register (this clears the TXE flag).
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPI_DR register (this clears the RXNE flag).

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 are 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 (flag TCIF 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 TXE=1 and then until BSY=0.

Note: During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, it is mandatory to wait first until TXE=1 and then until BSY=0 after writing the last data.

Figure 231. Transmission using DMA

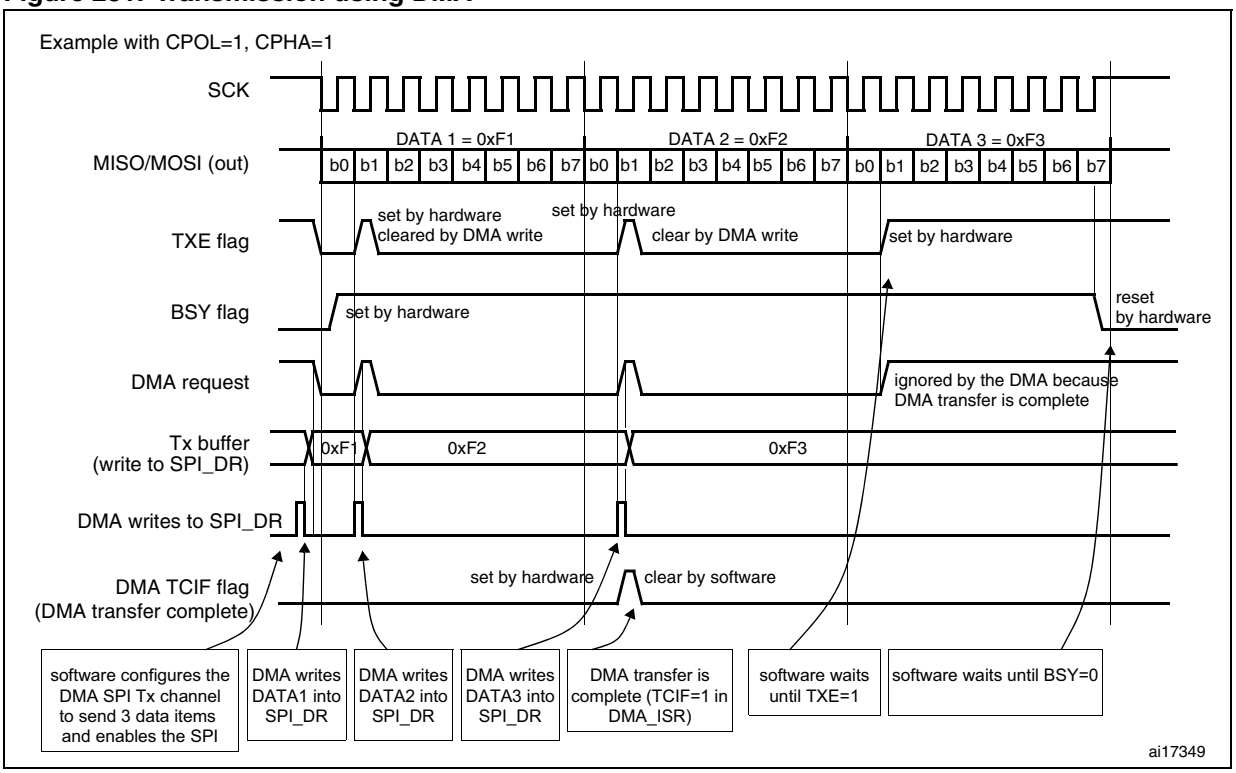
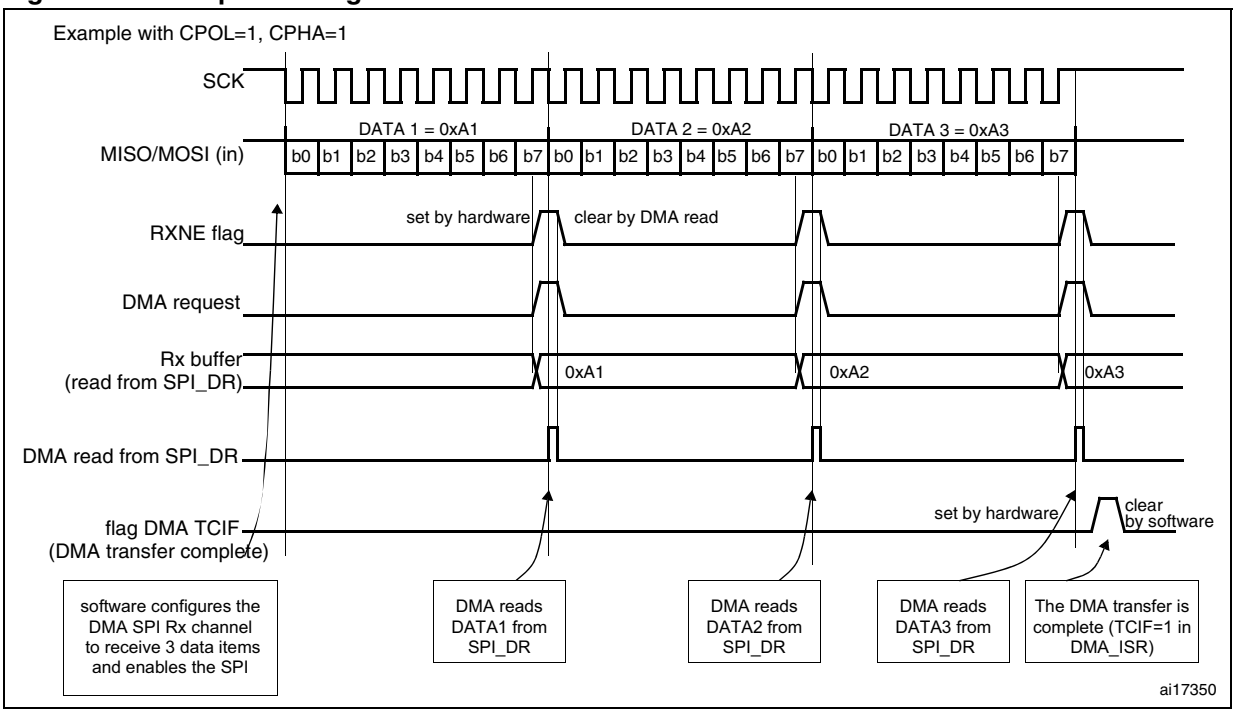


Figure 232. Reception using DMA



DMA capability with CRC

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication are automatic that is without using the bit CRCNEXT. After the CRC reception, the CRC must be read in the SPI_DR register to clear the RXNE flag.

At the end of data and CRC transfers, the CRCERR flag in SPI_SR is set if corruption occurs during the transfer.

21.3.10 Error flags

Master mode fault (MODF)

Master mode fault occurs when the master device has its NSS pin pulled low (in NSS hardware mode) or SSI bit low (in NSS software mode), this automatically sets the MODF bit. Master mode fault affects the SPI peripheral 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 SPI_SR register while the MODF bit is set.
2. Then write to the SPI_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 setting of the SPE and MSTR bits while the MODF bit is set.

In a slave device the MODF bit cannot be set. However, in a multimaster configuration, the device can be in slave mode with this MODF bit set. In this case, the MODF bit indicates that there might have been a multimaster conflict for system control. An interrupt routine can be used to recover cleanly from this state by performing a reset or returning to a default state.

Overrun condition

An overrun condition occurs when the master device has sent data bytes and the slave device has not cleared the RXNE bit resulting from the previous data byte transmitted. When an overrun condition occurs:

- the OVR bit is set and an interrupt is generated if the ERRIE bit is set.

In this case, the receiver buffer contents will not be updated with the newly received data from the master device. A read from the SPI_DR register returns this byte. All other subsequently transmitted bytes are lost.

Clearing the OVR bit is done by a read from the SPI_DR register followed by a read access to the SPI_SR register.

CRC error

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CR1 register is set. The CRCERR flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRCR value.

21.3.11 SPI interrupts

Table 118. SPI interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
CRC error flag	CRCERR	

21.4 SPI 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).

21.4.1 SPI control register 1 (SPI_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDIMODE:** Bidirectional data mode enable
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:** CRC transfer next
0: Data phase (no CRC phase)
1: Next transfer is CRC (CRC phase)
Note: When the SPI is configured in full duplex or transmitter only modes, CRCNEXT must be written as soon as the last data is written to the SPI_DR register.
When the SPI is configured in receiver only mode, CRCNEXT must be set after the second last data reception.
This bit should be kept cleared when the transfers are managed by DMA.

Bit 11 **DFF:** Data frame format
0: 8-bit data frame format is selected for transmission/reception
1: 16-bit data frame format is selected for transmission/reception
Note: This bit should be written only when SPI is disabled (SPE = '0) for correct operation

Bit 10 **RXONLY:** Receive only
This bit combined with the BIDImode bit selects the direction of transfer in 2-line unidirectional mode. This bit is also useful in a multislave 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

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 IO value of the NSS pin is ignored.

Bit 7 **LSBFIRST:** Frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit should not be changed when communication is ongoing.

Bit 6 **SPE:** SPI enable
0: Peripheral disabled
1: Peripheral enabled
Note: When disabling the SPI, follow the procedure described in Section 21.3.8: Disabling the SPI.

Bits 5:3 **BR[2:0]:** Baud rate control
000: $f_{PCLK}/2$
001: $f_{PCLK}/4$
010: $f_{PCLK}/8$
011: $f_{PCLK}/16$
100: $f_{PCLK}/32$
101: $f_{PCLK}/64$
110: $f_{PCLK}/128$
111: $f_{PCLK}/256$
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.

Bit1 **CPOL:** Clock polarity
0: CK to 0 when idle
1: CK to 1 when idle
Note: This bit should not be changed when communication is ongoing.

Bit 0 **CPHA:** Clock phase
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
Note: This bit should not be changed when communication is ongoing.

21.4.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXEIE	RXNEIE	ERRIE	Res.	Res.	SSOE	TXDMAEN	RXDMAEN
								rw	rw	rw			rw	rw	rw

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

Bit 7 **TXEIE:** Tx buffer empty interrupt enable
0: TXE interrupt masked
1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable
0: RXNE interrupt masked
1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable
This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode).
0: Error interrupt is masked
1: Error interrupt is enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SSOE:** SS output enable
0: SS output is disabled in master mode and the cell can work in multimaster configuration
1: SS output is enabled in master mode and when the cell is enabled. The cell cannot work in a multimaster environment.

Bit 1 **TXDMAEN:** Tx buffer DMA enable
When this bit is set, the DMA request is made 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, the DMA request is made whenever the RXNE flag is set.
0: Rx buffer DMA disabled
1: Rx buffer DMA enabled

21.4.3 SPI status register (SPI_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BSY	OVR	MODF	CRC ERR	Reserved		TXE	RXNE
								r	r	r	rc_w0			r	r

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

Bit 7 **BSY:** Busy flag

0: SPInot busy
1: SPIis busy in communication or Tx buffer is not empty
This flag is set and cleared by hardware.

Note: BSY flag must be used with caution: refer to Section 21.3.7: Status flags and Section 21.3.8: Disabling the SPI.

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 21.3.10 on page 533* for the software sequence.

Bit 4 **CRCERR:** CRC error flag

0: CRC value received matches the SPI_RXCRCR value
1: CRC value received does not match the SPI_RXCRCR value
This flag is set by hardware and cleared by software writing 0.

Bits 3:2 Reserved

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

21.4.4 SPI data register (SPI_DR)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DR[15:0]:** Data register

Data received or to be transmitted.

The data register is split into 2 buffers - one for writing (Transmit Buffer) and another one for reading (Receive buffer). A write to the data register will write into the Tx buffer and a read from the data register will return the value held in the Rx buffer.

Notes for the SPI mode:

Depending on the data frame format selection bit (DFF in SPI_CR1 register), the data sent or received is either 8-bit or 16-bit. This selection has to be made before enabling the SPI to ensure correct operation.

For an 8-bit data frame, the buffers are 8-bit and only the LSB of the register (SPI_DR[7:0]) is used for transmission/reception. When in reception mode, the MSB of the register (SPI_DR[15:8]) is forced to 0.

For a 16-bit data frame, the buffers are 16-bit and the entire register, SPI_DR[15:0] is used for transmission/reception.

21.4.5 SPI CRC polynomial register (SPI_CRCPR)

Address offset: 0x10

Reset value: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0007h) is the reset value of this register. Another polynomial can be configured as required.

21.4.6 SPI RX CRC register (SPI_RXCRCR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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 SPI_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_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 data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY Flag is set could return an incorrect value.

21.4.7 SPI TX CRC register (SPI_TXCRCR)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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 SPI_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_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 data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set could return an incorrect value.

21.4.8 SPI register map

The table provides shows the SPI register map and reset values.

Table 119. SPI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPI_CR1**	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPI_CR2**	Reserved																								TXEIE	RXNEIE	ERRIE	Reserved		SSOE	TXDMAEN	RXDMAEN
	Reset value																									0	0	0			0	0	0
0x08	**SPI_SR**	Reserved																								BSY	OVR	MODF	CRCERR	Reserved		TXE	RXNE
	Reset value																									0	0	0	0			1	0
0x0C	**SPI_DR**	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPI_CRCPR**	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPI_RXCRCR**	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPI_TXCRCR**	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to for the register boundary addresses.

22 Inter-integrated circuit (I2C) interface

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

22.1 I2C introduction

I2C (inter-integrated circuit) bus Interface serves as an interface between the microcontroller and the serial I2C bus. It provides multimaster capability, and controls all I2C bus-specific sequencing, protocol, arbitration and timing. It supports standard and fast speed modes. It is also SMBus 2.0 compatible.

It may be used for a variety of purposes, including CRC generation and verification, SMBus (system management bus) and PMBus (power management bus).

Depending on specific device implementation DMA capability can be available for reduced CPU overload.

22.2 I2C main features

- Parallel-bus/I2C protocol converter
- Multimaster capability: the same interface can act as Master or Slave
- I2C Master features:
  - Clock generation
  - Start and Stop generation
- I2C Slave features:
  - Programmable I2C Address detection
  - Dual Addressing Capability to acknowledge 2 slave addresses
  - Stop bit detection
- Generation and detection of 7-bit/10-bit addressing and General Call
- Supports different communication speeds:
  - Standard Speed (up to 100 kHz),
  - Fast Speed (up to 400 kHz)
- Status flags:
  - Transmitter/Receiver mode flag
  - End-of-Byte transmission flag
  - I2C busy flag
- Error flags:
  - Arbitration lost condition for master mode

  - Acknowledgement failure after address/ data transmission
  - Detection of misplaced start or stop condition
  - Overrun/Underrun if clock stretching is disabled
- 2 Interrupt vectors:
  - 1 Interrupt for successful address/ data communication
  - 1 Interrupt for error condition
- Optional clock stretching
- 1-byte buffer with DMA capability
- Configurable PEC (packet error checking) generation or verification:
  - PEC value can be transmitted as last byte in Tx mode
  - PEC error checking for last received byte
- SMBus 2.0 Compatibility:
  - 25 ms clock low timeout delay
  - 10 ms master cumulative clock low extend time
  - 25 ms slave cumulative clock low extend time
  - Hardware PEC generation/verification with ACK control
  - Address Resolution Protocol (ARP) supported
- PMBus Compatibility

Note: Some of the above features may not be available in certain products. The user should refer to the product data sheet, to identify the specific features supported by the I^2C interface implementation.

22.3 I^2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz) or fast (up to 400 kHz) I^2C bus.

22.3.1 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a Stop generation occurs, allowing multimaster capability.

Communication flow

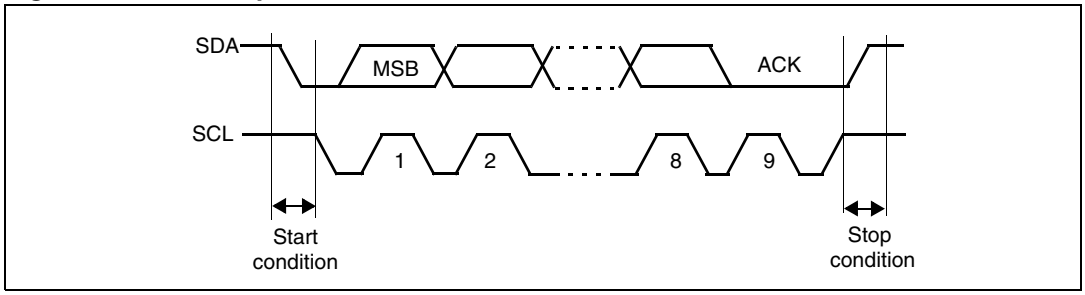
In Master mode, the I^2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to *Figure 233*.

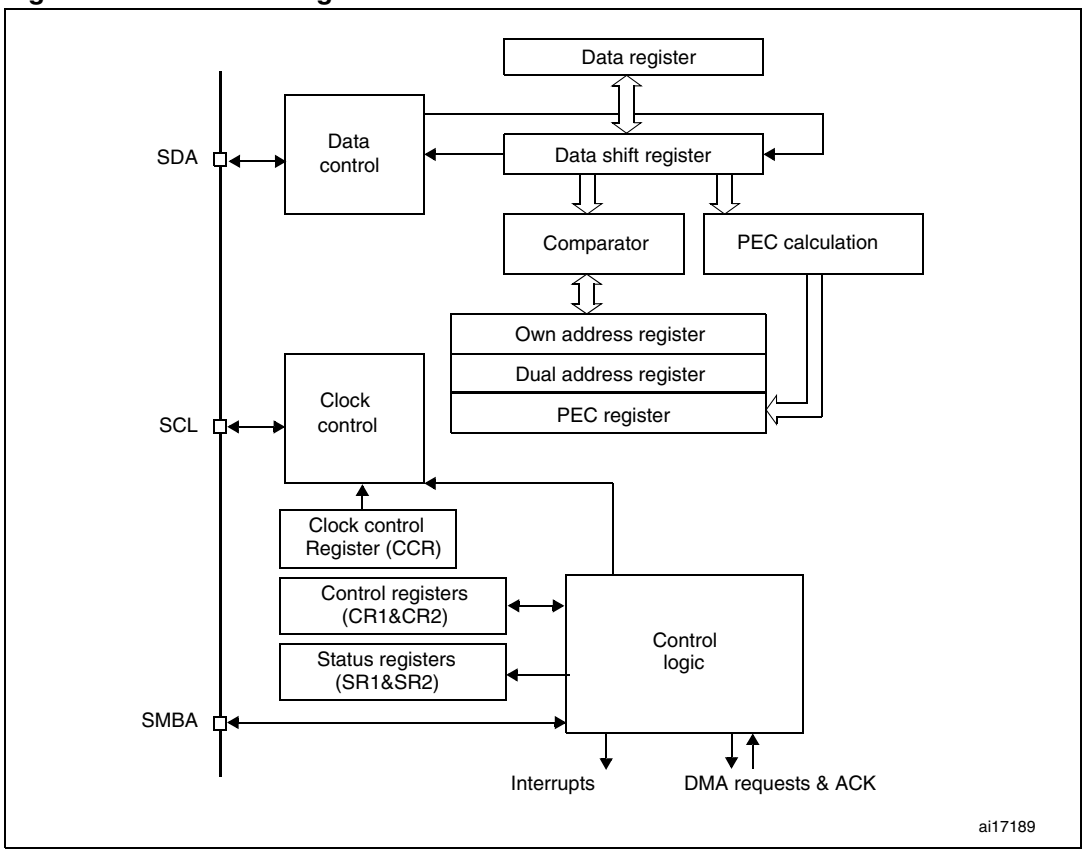
Figure 233. I^2C bus protocol



Acknowledge may be enabled or disabled by software. The I^2C interface addresses (dual addressing 7-bit/ 10-bit and/or general call address) can be selected by software.

The block diagram of the I^2C interface is shown in *Figure 234*.

Figure 234. I²C block diagram



1. SMBA is an optional signal in SMBus mode. This signal is not applicable if SMBus is disabled.

22.3.2 I²C slave mode

By default the I²C interface operates in Slave mode. To switch from default Slave mode to Master mode a Start condition generation is needed.

The peripheral input clock must be programmed in the I2C_CR2 register in order to generate correct timings. The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of the interface (OAR1) and with OAR2 (if ENDUAL=1) or the General Call address (if ENGC = 1).

Note: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.

Header or address not matched: the interface ignores it and waits for another Start condition.

Header matched (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set and waits for the 8-bit slave address.

Address matched: the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
- If ENDUAL=1, the software has to read the DUALF bit to check which slave address has been acknowledged.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

The TRA bit indicates whether the slave is in Receiver or Transmitter mode.

Slave transmitter

Following the address reception and after clearing ADDR, the slave sends bytes from the DR register to the SDA line via the internal shift register.

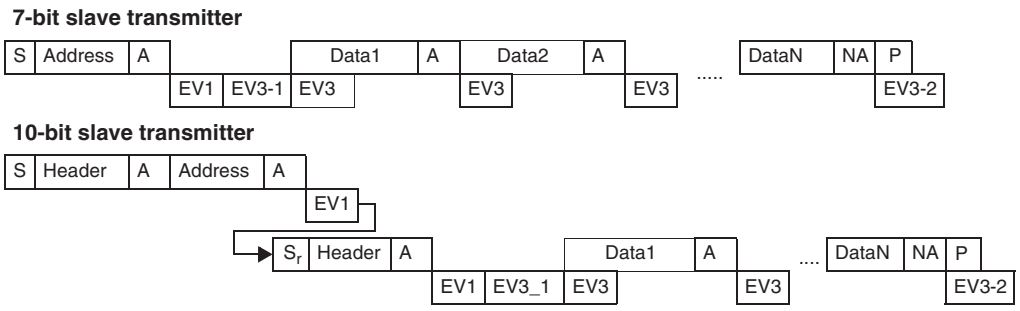
The slave stretches SCL low until ADDR is cleared and DR filled with the data to be sent (see *Figure 235* Transfer sequencing EV1 EV3).

When the acknowledge pulse is received:

- The TxE bit is set by hardware with an interrupt if the ITEVFEN and the ITBUFEN bits are set.

If TxE is set and some data were not written in the I2C_DR register before the end of the next data transmission, the BTF bit is set and the interface waits until BTF is cleared by a read to I2C_SR1 followed by a write to the I2C_DR register, stretching SCL low.

Figure 235. Transfer sequence diagram for slave transmitter



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV1: ADDR=1, cleared by reading SR1 followed by reading SR2
EV3-1: TxE=1, shift register empty, data register empty, write Data1 in DR.
EV3: TxE=1, shift register not empty, data register empty, cleared by writing DR
EV3-2: AF=1; AF is cleared by writing '0' in AF bit of SR1 register.

ai18209

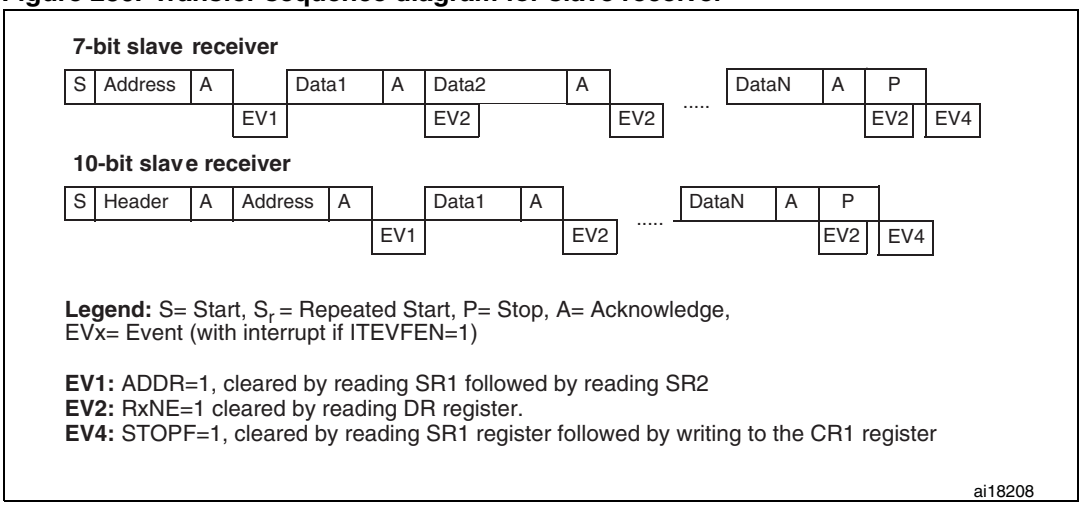
Slave receiver

Following the address reception and after clearing ADDR, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RxNE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bit is set.

If RxNE is set and the data in the DR register is not read before the end of the next data reception, the BTF bit is set and the interface waits until BTF is cleared by a read from I2C_SR1 followed by a read from the I2C_DR register, stretching SCL low (see *Figure 236* Transfer sequencing).

Figure 236. Transfer sequence diagram for slave receiver



1. The EV1 event stretches SCL low until the end of the corresponding software sequence.
2. The EV2 software sequence must be completed before the end of the current byte transfer
3. After checking the SR1 register content, the user should perform the complete clearing sequence for each flag found set.
   Thus, for ADDR and STOPF flags, the following sequence is required inside the I2C interrupt routine:
   READ SR1
   if (ADDR == 1) {READ SR1; READ SR2}
   if (STOPF == 1) {READ SR1; WRITE CR1}
   The purpose is to make sure that both ADDR and STOPF flags are cleared if both are found set.

Closing slave communication

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets:

- The STOPF bit and generates an interrupt if the ITEVFEN bit is set.

The STOPF bit is cleared by a read of the SR1 register followed by a write to the CR1 register (see *Figure 236: Transfer sequence diagram for slave receiver* EV4).

22.3.3 I²C master mode

In Master mode, the I²C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition.

Master mode is selected as soon as the Start condition is generated on the bus with a START bit.

The following is the required sequence in master mode.

- Program the peripheral input clock in I2C_CR2 Register in order to generate correct timings
- Configure the clock control registers
- Configure the rise time register
- Program the I2C_CR1 register to enable the peripheral
- Set the START bit in the I2C_CR1 register to generate a Start condition

The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

Start condition

Setting the START bit causes the interface to generate a Start condition and to switch to Master mode (M/SL bit set) when the BUSY bit is cleared.

Note: In master mode, setting the START bit causes the interface to generate a ReStart condition at the end of the current byte transfer.

Once the Start condition is sent:

- The SB bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address (see *Figure 237* & *Figure 238* Transfer sequencing EV5).

Slave address transmission

Then the slave address is sent to the SDA line via the internal shift register.

- In 10-bit addressing mode, sending the header sequence causes the following event:
  - The ADD10 bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a write in the DR register with the second address byte (see *Figure 237* & *Figure 238* Transfer sequencing).
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 237* & *Figure 238* Transfer sequencing).
- In 7-bit addressing mode, one address byte is sent.

  As soon as the address byte is sent,
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 237* & *Figure 238* Transfer sequencing).

The master can decide to enter Transmitter or Receiver mode depending on the LSB of the slave address sent.

- In 7-bit addressing mode,
  - To enter Transmitter mode, a master sends the slave address with LSB reset.
  - To enter Receiver mode, a master sends the slave address with LSB set.
- In 10-bit addressing mode,
  - To enter Transmitter mode, a master sends the header (11110xx0) and then the slave address, (where xx denotes the two most significant bits of the address).
  - To enter Receiver mode, a master sends the header (11110xx0) and then the slave address. Then it should send a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the address).

  The TRA bit indicates whether the master is in Receiver or Transmitter mode.

Master transmitter

Following the address transmission and after clearing ADDR, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits until the first data byte is written into I2C_DR (see *Figure 237* Transfer sequencing EV8_1).

When the acknowledge pulse is received, the TxE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set.

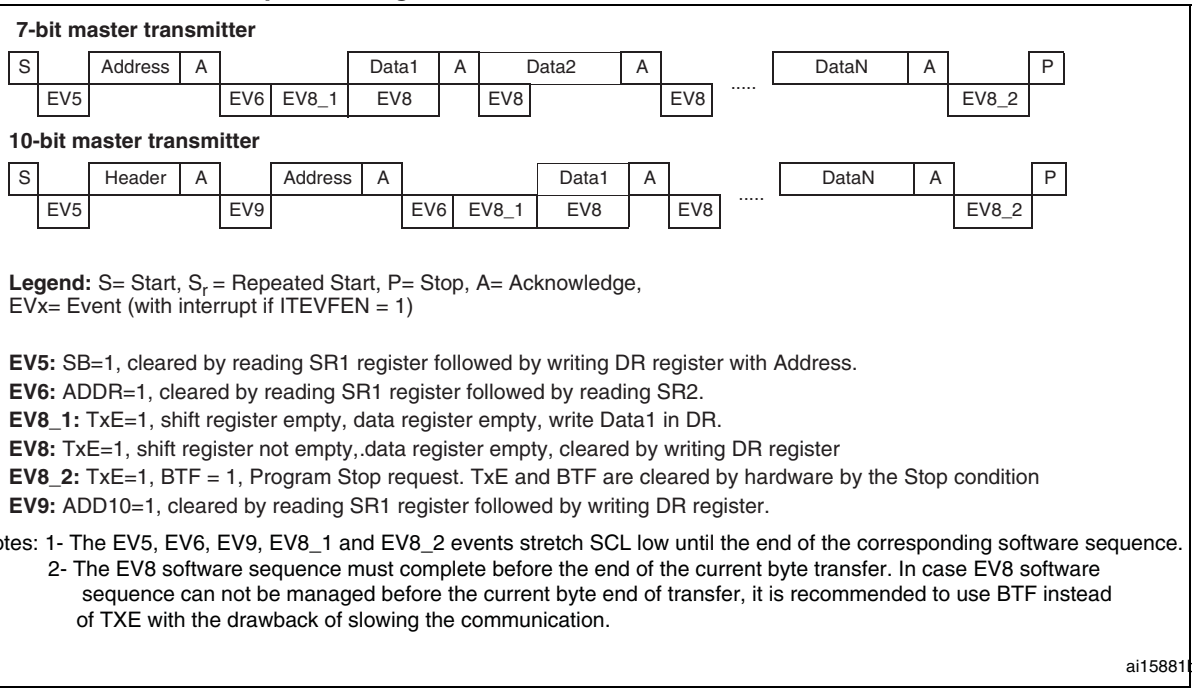
If TxE is set and a data byte was not written in the DR register before the end of the last data transmission, BTF is set and the interface waits until BTF is cleared by a read from I2C_SR1 followed by a write to I2C_DR, stretching SCL low.

Closing the communication

After the last byte is written to the DR register, the STOP bit is set by software to generate a Stop condition (see *Figure 237* Transfer sequencing EV8_2). The interface automatically goes back to slave mode (M/SL bit cleared).

Note: Stop condition should be programmed during EV8_2 event, when either TxE or BTF is set.

Figure 237. Transfer sequence diagram for master transmitter



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge,
EVx= Event (with interrupt if ITEVFEN = 1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register with Address.
EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2.
EV8_1: TxE=1, shift register empty, data register empty, write Data1 in DR.
EV8: TxE=1, shift register not empty,.data register empty, cleared by writing DR register
EV8_2: TxE=1, BTF = 1, Program Stop request. TxE and BTF are cleared by hardware by the Stop condition
EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

Notes: 1- The EV5, EV6, EV9, EV8_1 and EV8_2 events stretch SCL low until the end of the corresponding software sequence.
2- The EV8 software sequence must complete before the end of the current byte transfer. In case EV8 software sequence can not be managed before the current byte end of transfer, it is recommended to use BTF instead of TXE with the drawback of slowing the communication.

ai15881b

Master receiver

Following the address transmission and after clearing ADDR, the I^2C interface enters Master Receiver mode. In this mode the interface receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

1. An acknowledge pulse if the ACK bit is set
2. The RxNE bit is set and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set (see *Figure 238* Transfer sequencing EV7).

If the RxNE bit is set and the data in the DR register is not read before the end of the last data reception, the BTF bit is set by hardware and the interface waits until BTF is cleared by a read in the DR register, stretching SCL low.

Closing the communication

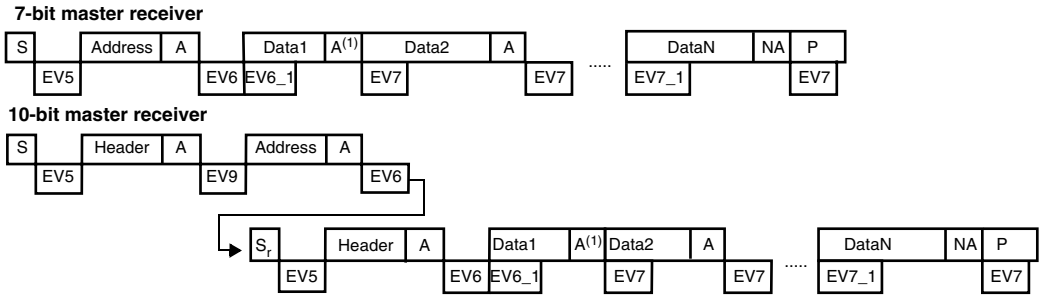
Method 1: This method is for the case when the I2C is used with interrupts that have the highest priority in the application.

The master sends a NACK for the last byte received from the slave. After receiving this NACK, the slave releases the control of the SCL and SDA lines. Then the master can send a Stop/Restart condition.

1. To generate the nonacknowledge pulse after the last received data byte, the ACK bit must be cleared just after reading the second last data byte (after second last RxNE event).
2. To generate the Stop/Restart condition, software must set the STOP/START bit just after reading the second last data byte (after the second last RxNE event).
3. In case a single byte has to be received, the Acknowledge disable and the Stop condition generation are made just after EV6 (in EV6_1, just after ADDR is cleared).

After the Stop condition generation, the interface goes automatically back to slave mode (M/SL bit cleared).

Figure 238. Method 1: transfer sequence diagram for master receiver



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register.

EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2. In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.

EV6_1: no associated flag event, used for 1 byte reception only. The Acknowledge disable and Stop condition generation are made just after EV6, that is after ADDR is cleared.

EV7: RxNE=1 cleared by reading DR register.

EV7_1: RxNE=1 cleared by reading DR register, program ACK=0 and STOP request

EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

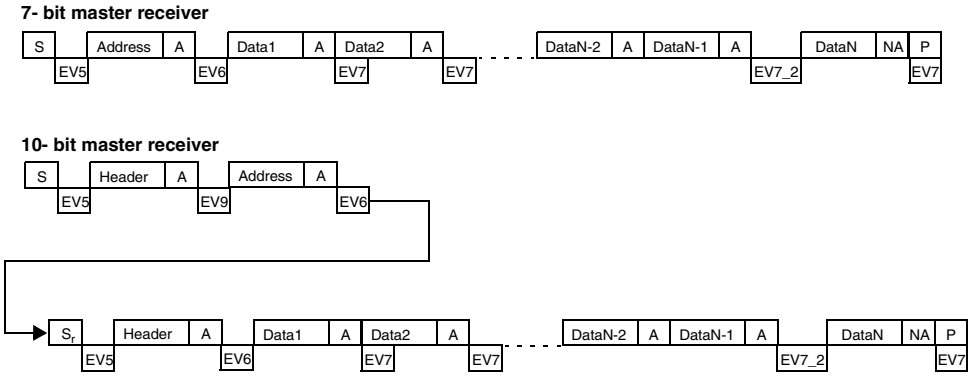
ai15882

1. If a single byte is received, it is NA.
2. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. The EV7 software sequence must complete before the end of the current byte transfer. In case EV7 software sequence can not be managed before the current byte end of transfer, it is recommended to use BTF instead of RXNE with the drawback of slowing the communication.
4. The EV6_1 or EV7_1 software sequence must complete before the ACK pulse of the current byte transfer.

Method 2: This method is for the case when the I2C is used with interrupts that do not have the highest priority in the application or when the I2C is used with polling.

With this method, DataN_2 is not read, so that after DataN_1, the communication is stretched (both RxNE and BTF are set). Then, clear the ACK bit before reading DataN-2 in DR to ensure it is be cleared before the DataN Acknowledge pulse. After that, just after reading DataN_2, set the STOP/ START bit and read DataN_1. After RxNE is set, read DataN. This is illustrated below:

Figure 239. Method 2: transfer sequence diagram for master receiver when N>2



Legend: S = Start, S_r = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge,
EVx = Event (with interrupt if ITEVFEN = 1)
EV5: SB=1, cleared by reading SR1 register followed by writing the DR register.
EV6: ADDR1, cleared by reading SR1 register followed by reading SR2.
In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.
EV7: RxNE=1, cleared by reading DR register
EV7_2: BTF = 1, DataN-2 in DR and DataN-1 in shift register, program ACK = 0, Read DataN-2 in DR.
Program STOP = 1, read DataN-1.
EV9: ADD10= 1, cleared by reading SR1 register followed by writing DR register.

1. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
2. The EV7 software sequence must complete before the end of the current byte transfer.In case EV7 software sequence can not be managed before the current byte end of transfer, it is recommended to use BTF instead of RXNE with the drawback of slowing the communication.

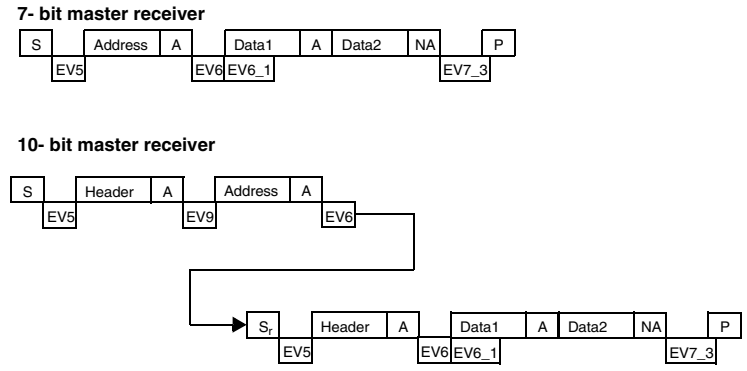
When 3 bytes remain to be read:

- RxNE = 1 => Nothing (DataN-2 not read).
- DataN-1 received
- BTF = 1 because both shift and data registers are full: DataN-2 in DR and DataN-1 in the shift register => SCL tied low: no other data will be received on the bus.
- Clear ACK bit
- Read DataN-2 in DR => This will launch the DataN reception in the shift register
- DataN received (with a NACK)
- Program START/STOP
- Read DataN-1
- RxNE = 1
- Read DataN

The procedure described above is valid for N>2. The cases where a single byte or two bytes are to be received should be handled differently, as described below:

- Case of a single byte to be received:
  - In the ADDR event, clear the ACK bit.
  - Clear ADDR
  - Program the STOP/START bit.
  - Read the data after the RxNE flag is set.
- Case of two bytes to be received:
  - Set POS and ACK
  - Wait for the ADDR flag to be set
  - Clear ADDR
  - Clear ACK
  - Wait for BTF to be set
  - Program STOP
  - Read DR twice

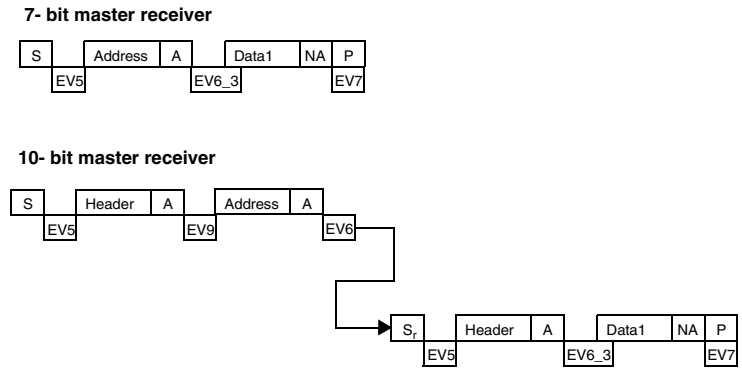
Figure 240. Method 2: transfer sequence diagram for master receiver when N=2



Legend: S = Start, S_r = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge,
EVx = Event (with interrupt if ITEVFEN = 1)
EV5: SB=1, cleared by reading SR1 register followed by writing the DR register.
EV6: ADDR1, cleared by reading SR1 register followed by reading SR2.
In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.
EV6_1: No associated flag event. The acknowledge disable should be done just after EV6, that is after ADDR is cleared.
EV7_3: BTF = 1, program STOP = 1, read DR twice (Read Data1 and Data2) just after programming the STOP.
EV9: ADD10= 1, cleared by reading SR1 register followed by writing DR register.

1. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
2. The EV6_1 software sequence must complete before the ACK pulse of the current byte transfer.

Figure 241. Method 2: transfer sequence diagram for master receiver when N=1



Legend: S = Start, S_r = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge, EVx = Event (with interrupt if ITEVFEN = 1)
EV5: SB=1, cleared by reading SR1 register followed by writing the DR register.
EV6: ADDR =1, cleared by reading SR1 resister followed by reading SR2 register.
EV6_3: ADDR = 1, program ACK = 0, clear ADDR by reading SR1 register followed by reading SR2 register, program STOP =1 just after ADDR is cleared.
Note: The EV6_3 software sequence must complete before the current byte end of transfer.
EV7: RxNE =1, cleared by reading DR register.
EV9: ADD10= 1, cleared by reading SR1 register followed by writing DR register.

1. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.

22.3.4 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

This error occurs when the I^2C interface detects an external Stop or Start condition during an address or a data transfer. In this case:

- the BERR bit is set and an interrupt is generated if the ITERREN bit is set
- in Slave mode: data are discarded and the lines are released by hardware:
  - in case of a misplaced Start, the slave considers it is a restart and waits for an address, or a Stop condition
  - in case of a misplaced Stop, the slave behaves like for a Stop condition and the lines are released by hardware
- In Master mode: the lines are not released and the state of the current transmission is not affected. It is up to the software to abort or not the current transmission

Acknowledge failure (AF)

This error occurs when the interface detects a nonacknowledge bit. In this case:

- the AF bit is set and an interrupt is generated if the ITERREN bit is set
- a transmitter which receives a NACK must reset the communication:
  - If Slave: lines are released by hardware
  - If Master: a Stop or repeated Start condition must be generated by software

Arbitration lost (ARLO)

This error occurs when the I^2C interface detects an arbitration lost condition. In this case,

- the ARLO bit is set by hardware (and an interrupt is generated if the ITERREN bit is set)
- the I^2C Interface goes automatically back to slave mode (the M/SL bit is cleared). When the I^2C loses the arbitration, it is not able to acknowledge its slave address in the same transfer, but it can acknowledge it after a repeated Start from the winning master.
- lines are released by hardware

Overrun/underrun error (OVR)

An overrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is receiving data. The interface has received a byte (RxNE=1) and the data in DR has not been read, before the next byte is received by the interface. In this case,

- The last received byte is lost.
- In case of Overrun error, software should clear the RxNE bit and the transmitter should re-transmit the last received byte.

Underrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is transmitting data. The interface has not updated the DR with the next byte (TxE=1), before the clock comes for the next byte. In this case,

- The same byte in the DR register will be sent again
- The user should make sure that data received on the receiver side during an underrun error are discarded and that the next bytes are written within the clock low time specified in the I^2C bus standard.

For the first byte to be transmitted, the DR must be written after ADDR is cleared and before the first SCL rising edge. If not possible, the receiver must discard the first data.

22.3.5 SDA/SCL line control

- If clock stretching is enabled:
  - Transmitter mode: If TxE=1 and BTF=1: the interface holds the clock line low before transmission to wait for the microcontroller to read SR1 and then write the byte in the Data Register (both buffer and shift register are empty).
  - Receiver mode: If RxNE=1 and BTF=1: the interface holds the clock line low after reception to wait for the microcontroller to read SR1 and then read the byte in the Data Register (both buffer and shift register are full).
- If clock stretching is disabled in Slave mode:
  - Overrun Error in case of RxNE=1 and no read of DR has been done before the next byte is received. The last received byte is lost.
  - Underrun Error in case TxE=1 and no write into DR has been done before the next byte must be transmitted. The same byte will be sent again.
  - Write Collision not managed.

22.3.6 SMBus

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of toggling individual control lines.

The System Management Bus Specification refers to three types of devices. A *slave* is a device that is receiving or responding to a command. A *master* is a device that issues commands, generates the clocks, and terminates the transfer. A *host* is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

Similarities between SMBus and I^2C

- 2 wire bus protocol (1 Clk, 1 Data) + SMBus Alert line optional
- Master-slave communication, Master provides clock
- Multi master capability
- SMBus data format similar to I^2C 7-bit addressing format (*Figure 233*).

Differences between SMBus and I^2C

The following table describes the differences between SMBus and I^2C.

Table 120. SMBus vs. I^2C

SMBus	I^2C
Max. speed 100 kHz	Max. speed 400 kHz
Min. clock speed 10 kHz	No minimum clock speed
35 ms clock low timeout	No timeout
Logic levels are fixed	Logic levels are V_{DD} dependent
Different address types (reserved, dynamic etc.)	7-bit, 10-bit and general call slave address types
Different bus protocols (quick command, process call etc.)	No bus protocols

SMBus application usage

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.

Device identification

Any device that exists on the System Management Bus as a slave has a unique address called the Slave Address. For the list of reserved slave addresses, refer to the SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

Bus protocols

The SMBus specification supports up to 9 bus protocols. For more details of these protocols and SMBus address types, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*). These protocols should be implemented by the user software.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following attributes:

- Address assignment uses the standard SMBus physical layer arbitration mechanism
- Assigned addresses remain constant while device power is applied; address retention through device power loss is also allowed
- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
- Any SMBus master can enumerate the bus

Unique device identifier (UDID)

In order to provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID).

For the details on 128 bit UDID and more information on ARP, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

SMBus alert mode

SMBus Alert is an optional signal with an interrupt line for devices that want to trade their ability to master for a pin. SMBA is a wired-AND signal just as the SCL and SDA signals are. SMBA is used in conjunction with the SMBus General Call Address. Messages invoked with the SMBus are 2 bytes long.

A slave-only device can signal the host through SMBA that it wants to talk by setting ALERT bit in I2C_CR1 register. The host processes the interrupt and simultaneously accesses all SMBA devices through the *Alert Response Address* (known as ARA having a value 0001 100X). Only the device(s) which pulled SMBA low will acknowledge the Alert Response Address. This status is identified using SMBALERT Status flag in I2C_SR1 register. The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBA low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address the device must disengage its SMBA pull-down. If the host still sees SMBA low when the message transfer is complete, it knows to read the ARA again.
A host which does not implement the SMBA signal may periodically access the ARA.

For more details on SMBus Alert mode, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

Timeout error

There are differences in the timing specifications between I²C and SMBus.
SMBus defines a clock low timeout, TIMEOUT of 35 ms. Also SMBus specifies TLOW: SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW: MEXT as the cumulative clock low extend time for a master device. For more details on these timeouts, refer to SMBus specification ver. 2.0 (*http://smbus.org/specs/*).

The status flag Timeout or Tlow Error in I2C_SR1 shows the status of this feature.

How to use the interface in SMBus mode

To switch from I²C mode to SMBus mode, the following sequence should be performed.

- Set the SMBus bit in the I2C_CR1 register
- Configure the SMBTYPE and ENARP bits in the I2C_CR1 register as required for the application

If you want to configure the device as a master, follow the Start condition generation procedure in *Section 22.3.3: I2C master mode*. Otherwise, follow the sequence in *Section 22.3.2: I2C slave mode*.

The application has to control the various SMBus protocols by software.

- SMB Device Default Address acknowledged if ENARP=1 and SMBTYPE=0
- SMB Host Header acknowledged if ENARP=1 and SMBTYPE=1
- SMB Alert Response Address acknowledged if SMBALERT=1

22.3.7 DMA requests

DMA requests (when enabled) are generated only for data transfer. DMA requests are generated by Data Register becoming empty in transmission and Data Register becoming full in reception. The DMA request must be served before the end of the current byte transfer. When the number of data transfers which has been programmed for the corresponding DMA channel is reached, the DMA controller sends an End of Transfer EOT signal to the I²C interface and generates a Transfer Complete interrupt if enabled:

- Master transmitter: In the interrupt routine after the EOT interrupt, disable DMA requests then wait for a BTF event before programming the Stop condition.
- Master receiver: when the number of bytes to be received is equal to or greater than two, the DMA controller sends a hardware signal, EOT_1, corresponding to the last but one data byte (number_of_bytes – 1). If, in the I2C_CR2 register, the LAST bit is set, I²C automatically sends a NACK after the next byte following EOT_1. The user can generate a Stop condition in the DMA Transfer Complete interrupt routine if enabled.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from a Memory area configured using the DMA peripheral (refer to the DMA specification) to the I2C_DR register whenever the TxE bit is set. To map a DMA channel for I²C transmission, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in the DMA_CPARx register. The data will be moved to this address from the memory after each TxE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded into I2C_DR from this memory after each TxE event.
3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each TxE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Set the DIR bit and, in the DMA_CCRx register, configure interrupts after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^2C interface and the DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for transmission.

Reception using DMA

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from the I2C_DR register to a Memory area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for I^2C reception, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in DMA_CPARx register. The data will be moved from this address to the memory after each RxNE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded from the I2C_DR register to this memory area after each RxNE event.
3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each RxNE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Reset the DIR bit and configure interrupts in the DMA_CCRx register after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^2C interface and DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for reception.

22.3.8 Packet error checking

A PEC calculator has been implemented to improve the reliability of communication. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial serially on each bit.

- PEC calculation is enabled by setting the ENPEC bit in the I2C_CR1 register. PEC is a CRC-8 calculated on all message bytes including addresses and R/W bits.
  - In transmission: set the PEC transfer bit in the I2C_CR1 register after the TxE event corresponding to the last byte. The PEC will be transferred after the last transmitted byte.
  - In reception: set the PEC bit in the I2C_CR1 register after the RxNE event corresponding to the last byte so that the receiver sends a NACK if the next received byte is not equal to the internally calculated PEC. In case of Master-Receiver, a NACK must follow the PEC whatever the check result.The PEC must be set before the ACK pulse of the current byte reception.
- A PECERR error flag/interrupt is also available in the I2C_SR1 register.
- If DMA and PEC calculation are both enabled:-
  - In transmission: when the I^2C interface receives an EOT signal from the DMA controller, it automatically sends a PEC after the last byte.
  - In reception: when the I^2C interface receives an EOT_1 signal from the DMA controller, it will automatically consider the next byte as a PEC and will check it. A DMA request is generated after PEC reception.
- To allow intermediate PEC transfers, a control bit is available in the I2C_CR2 register (LAST bit) to determine if it is really the last DMA transfer or not. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
- PEC calculation is corrupted by an arbitration loss.

22.4 I^2C interrupts

The table below gives the list of I^2C interrupt requests.

Table 121. I^2C Interrupt requests

Interrupt event	Event flag	Enable control bit
Start bit sent (Master)	SB	ITEVFEN
Address sent (Master) or Address matched (Slave)	ADDR	
10-bit header sent (Master)	ADD10	
Stop received (Slave)	STOPF	
Data byte transfer finished	BTF	
Receive buffer not empty	RxNE	ITEVFEN and ITBUFEN
Transmit buffer empty	TxE	

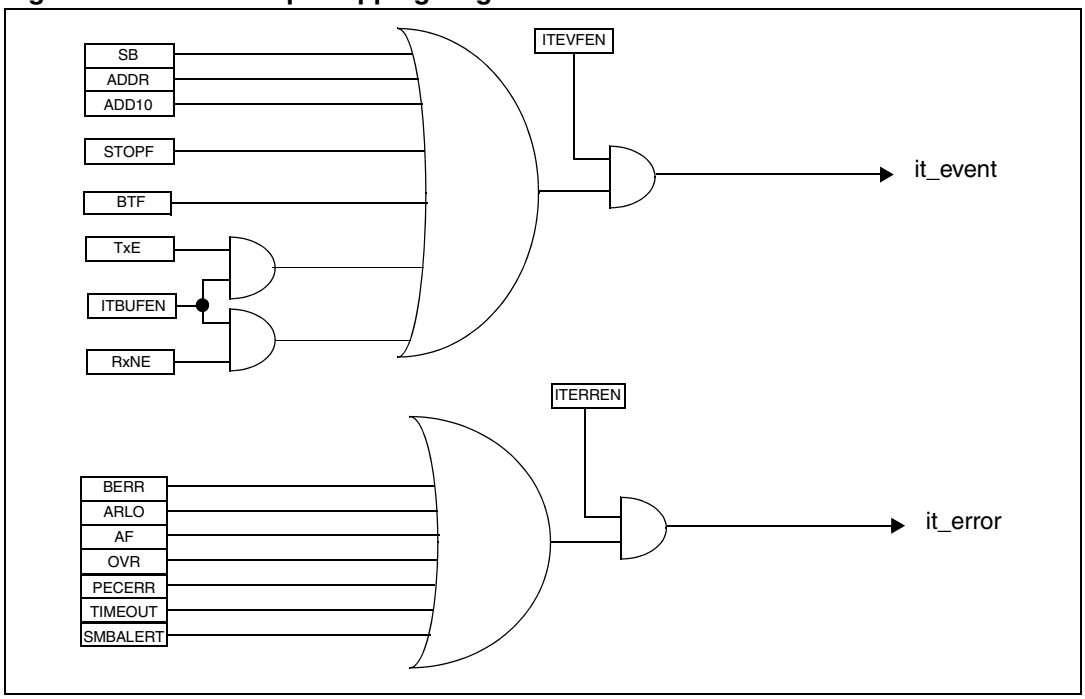
Table 121. I²C Interrupt requests (continued)

Interrupt event	Event flag	Enable control bit
Bus error	BERR	ITERREN
Arbitration loss (Master)	ARLO	
Acknowledge failure	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	
Timeout/Tlow error	TIMEOUT	
SMBus Alert	SMBALERT	

Note: 1 SB, ADDR, ADD10, STOPF, BTF, RxNE and TxE are logically ORed on the same interrupt channel.

2 BERR, ARLO, AF, OVR, PECERR, TIMEOUT and SMBALERT are logically ORed on the same interrupt channel.

Figure 242. I²C interrupt mapping diagram



22.5 I²C debug mode

When the microcontroller enters the debug mode (Cortex™-M3 core halted), the SMBUS timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module. For more details, refer to *Section 25.15.2: Debug support for timers, watchdog and I2C on page 658*.

22.6 I²C registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

22.6.1 I²C Control register 1 (I2C_CR1)

Address offset: 0x00
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYPE	Res.	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 15 **SWRST**: Software reset

When set, the I2C is under reset state. Before resetting this bit, make sure the I2C lines are released and the bus is free.

0: I²C Peripheral not under reset

1: I²C Peripheral under reset state

Note: This bit can be used in case the BUSY bit is set to ‘1 when no stop condition has been detected on the bus.

Bit 14 Reserved, forced by hardware to 0.

Bit 13 **ALERT**: SMBus alert

This bit is set and cleared by software, and cleared by hardware when PE=0.

0: Releases SMBA pin high. Alert Response Address Header followed by NACK.

1: Drives SMBA pin low. Alert Response Address Header followed by ACK.

Bit 12 **PEC**: Packet error checking

This bit is set and cleared by software, and cleared by hardware when PEC is transferred or by a START or Stop condition or when PE=0.

0: No PEC transfer

1: PEC transfer (in Tx or Rx mode)

Note: PEC calculation is corrupted by an arbitration loss.

Bit 11 **POS**: Acknowledge/PEC Position (for data reception)

This bit is set and cleared by software and cleared by hardware when PE=0.

0: ACK bit controls the (N)ACK of the current byte being received in the shift register. The PEC bit indicates that current byte in shift register is a PEC.
1: ACK bit controls the (N)ACK of the next byte which will be received in the shift register. The PEC bit indicates that the next byte in the shift register is a PEC

Note: The POS bit is used when the procedure for reception of 2 bytes (see Method 2: transfer sequence diagram for master receiver when N=2) is followed. It must be configured before data reception starts. In this case, to NACK the 2nd byte, the ACK bit must be cleared just after ADDR is cleared. To check the 2nd byte as PEC, the PEC bit must be set during the ADDR stretch event after configuring the POS bit.

Bit 10 **ACK**: Acknowledge enable

This bit is set and cleared by software and cleared by hardware when PE=0.
0: No acknowledge returned
1: Acknowledge returned after a byte is received (matched address or data)

Bit 9 **STOP**: Stop generation

The bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected.
In Master Mode:
0: No Stop generation.
1: Stop generation after the current byte transfer or after the current Start condition is sent.
In Slave mode:
0: No Stop generation.
1: Release the SCL and SDA lines after the current byte transfer.

Note: When the STOP, START or PEC bit is set, the software must not perform any write access to I2C_CR1 before this bit is cleared by hardware. Otherwise there is a risk of setting a second STOP, START or PEC request.

Bit 8 **START**: Start generation

This bit is set and cleared by software and cleared by hardware when start is sent or PE=0.
In Master Mode:
0: No Start generation
1: Repeated start generation
In Slave mode:
0: No Start generation
1: Start generation when the bus is free

Bit 7 **NOSTRETCH**: Clock stretching disable (Slave mode)

This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software.
0: Clock stretching enabled
1: Clock stretching disabled

Bit 6 **ENGC**: General call enable

0: General call disabled. Address 00h is NACKed.
1: General call enabled. Address 00h is ACKed.

Bit 5 **ENPEC:** PEC enable

0: PEC calculation disabled
1: PEC calculation enabled

Bit 4 **ENARP**: ARP enable
0: ARP disable
1: ARP enable
SMBus Device default address recognized if SMBTYPE=0
SMBus Host address recognized if SMBTYPE=1

Bit 3 **SMBTYPE**: SMBus type
0: SMBus Device
1: SMBus Host

Bit 2 Reserved, forced by hardware to 0.

Bit 1 **SMBUS**: SMBus mode
0: I^2C mode
1: SMBus mode

Bit 0 **PE**: Peripheral enable
0: Peripheral disable
1: Peripheral enable: the corresponding IOs are selected as alternate functions depending on SMBus bit.

Note: If this bit is reset while a communication is on going, the peripheral is disabled at the end of the current communication, when back to IDLE state.
All bit resets due to PE=0 occur at the end of the communication.
In master mode, this bit must not be reset before the end of the communication.

22.6.2 I^2C Control register 2 (I2C_CR2)

Address offset: 0x04
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMA EN	ITBUF EN	ITEVT EN	ITERR EN	Reserved		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, forced by hardware to 0.

Bit 12 **LAST**: DMA last transfer
0: Next DMA EOT is not the last transfer
1: Next DMA EOT is the last transfer

Note: This bit is used in master receiver mode to permit the generation of a NACK on the last received data.

Bit 11 **DMAEN**: DMA requests enable
0: DMA requests disabled
1: DMA request enabled when TxE=1 or RxNE =1

Bit 10 **ITBUFEN**: Buffer interrupt enable
0: TxE = 1 or RxNE = 1 does not generate any interrupt.
1:TxE = 1 or RxNE = 1 generates Event Interrupt (whatever the state of DMAEN)

Bit 9 **ITEVTEN**: Event interrupt enable

0: Event interrupt disabled
1: Event interrupt enabled

This interrupt is generated when:

–SB = 1 (Master)

–ADDR = 1 (Master/Slave)

–ADD10= 1 (Master)

–STOPF = 1 (Slave)

–BTF = 1 with no TxE or RxNE event

–TxE event to 1 if ITBUFEN = 1

–RxNE event to 1if ITBUFEN = 1

Bit 8 **ITERREN**: Error interrupt enable

0: Error interrupt disabled
1: Error interrupt enabled

This interrupt is generated when:

– BERR = 1
– ARLO = 1
– AF = 1
– OVR = 1
– PECERR = 1
– TIMEOUT = 1
– SMBALERT = 1

Bits 7:6 Reserved, forced by hardware to 0.

Bits 5:0 **FREQ[5:0]**: Peripheral clock frequency

The peripheral clock frequency must be configured using the input APB clock frequency (I2C peripheral connected to APB). The minimum allowed frequency is 2 MHz, the maximum frequency is limited by the maximum APB frequency (24 MHz) and an intrinsic limitation of 46 MHz.

0b000000: Not allowed
0b000001: Not allowed
0b000010: 2 MHz
...
0b011000: 24 MHz
Higher than 0b011000: Not allowed

22.6.3 I2C Own address register 1 (I2C_OAR1)

Address offset: 0x08
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **ADDMODE** Addressing mode (slave mode)
0: 7-bit slave address (10-bit address not acknowledged)
1: 10-bit slave address (7-bit address not acknowledged)

Bit 14 Should always be kept at 1 by software.

Bits 13:10 Reserved, forced by hardware to 0.

Bits 9:8 **ADD[9:8]**: Interface address
7-bit addressing mode: don't care
10-bit addressing mode: bits9:8 of address

Bits 7:1 **ADD[7:1]**: Interface address
bits 7:1 of address

Bit 0 **ADD0**: Interface address
7-bit addressing mode: don't care
10-bit addressing mode: bit 0 of address

22.6.4 I2C Own address register 2 (I2C_OAR2)

Address offset: 0x0C
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, forced by hardware to 0.

Bits 7:1 **ADD2[7:1]**: Interface address
bits 7:1 of address in dual addressing mode

Bit 0 **ENDUAL**: Dual addressing mode enable
0: Only OAR1 is recognized in 7-bit addressing mode
1: Both OAR1 and OAR2 are recognized in 7-bit addressing mode

22.6.5 I²C Data register (I2C_DR)

Address offset: 0x10
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, forced by hardware to 0.

Bits 7:0 **DR[7:0]** 8-bit data register

Byte received or to be transmitted to the bus.

–Transmitter mode: Byte transmission starts automatically when a byte is written in the DR register. A continuous transmit stream can be maintained if the next data to be transmitted is put in DR once the transmission is started (TxE=1)

–Receiver mode: Received byte is copied into DR (RxNE=1). A continuous transmit stream can be maintained if DR is read before the next data byte is received (RxNE=1).

Note: In slave mode, the address is not copied into DR.

Note: Write collision is not managed (DR can be written if TxE=0).

Note: If an ARLO event occurs on ACK pulse, the received byte is not copied into DR and so cannot be read.

22.6.6 I²C Status register 1 (I2C_SR1)

Address offset: 0x14
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bit 15 **SMBALERT**: SMBus alert

In SMBus host mode:
0: no SMBALERT
1: SMBALERT event occurred on pin
In SMBus slave mode:
0: no SMBALERT response address header
1: SMBALERT response address header to SMBALERT LOW received

– Cleared by software writing 0, or by hardware when PE=0.

Bit 14 **TIMEOUT**: Timeout or Tlow error

0: No timeout error
1: SCL remained LOW for 25 ms (Timeout)
or
Master cumulative clock low extend time more than 10 ms (Tlow:mext)
or
Slave cumulative clock low extend time more than 25 ms (Tlow:sext)

– When set in slave mode: slave resets the communication and lines are released by hardware
– When set in master mode: Stop condition sent by hardware
– Cleared by software writing 0, or by hardware when PE=0.

Note: This functionality is available only in SMBus mode.

Bit 13 Reserved, forced by hardware to 0.

Bit 12 **PECERR**: PEC Error in reception

0: no PEC error: receiver returns ACK after PEC reception (if ACK=1)

1: PEC error: receiver returns NACK after PEC reception (whatever ACK)

–Cleared by software writing 0, or by hardware when PE=0.

–

Bit 11 **OVR**: Overrun/Underrun

0: No overrun/underrun
1: Overrun or underrun

–Set by hardware in slave mode when NOSTRETCH=1 and:

–In reception when a new byte is received (including ACK pulse) and the DR register has not been read yet. New received byte is lost.

–In transmission when a new byte should be sent and the DR register has not been written yet. The same byte is sent twice.

–Cleared by software writing 0, or by hardware when PE=0.

Note: If the DR write occurs very close to SCL rising edge, the sent data is unspecified and a hold timing error occurs

Bit 10 **AF**: Acknowledge failure

0: No acknowledge failure
1: Acknowledge failure

–Set by hardware when no acknowledge is returned.

–Cleared by software writing 0, or by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost (master mode)

0: No Arbitration Lost detected
1: Arbitration Lost detected

Set by hardware when the interface loses the arbitration of the bus to another master

–Cleared by software writing 0, or by hardware when PE=0.

After an ARLO event the interface switches back automatically to Slave mode (M/SL=0).

Note: In SMBUS, the arbitration on the data in slave mode occurs only during the data phase, or the acknowledge transmission (not on the address acknowledge).

Bit 8 **BERR**: Bus error

0: No misplaced Start or Stop condition
1: Misplaced Start or Stop condition

–Set by hardware when the interface detects an SDA rising or falling edge while SCL is high, occurring in a non-valid position during a byte transfer.

–Cleared by software writing 0, or by hardware when PE=0.

Bit 7 **TxE**: Data register empty (transmitters)

0: Data register not empty
1: Data register empty

–Set when DR is empty in transmission. TxE is not set during address phase.

–Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0.

TxE is not set if either a NACK is received, or if next byte to be transmitted is PEC (PEC=1)

Note: TxE is not cleared by writing the first data being transmitted, or by writing data when BTF is set, as in both cases the data register is still empty.

Bit 6 **RxNE**: Data register not empty (receivers)

0: Data register empty
1: Data register not empty

–Set when data register is not empty in receiver mode. RxNE is not set during address phase.

–Cleared by software reading or writing the DR register or by hardware when PE=0.

RxNE is not set in case of ARLO event.

Note: RxNE is not cleared by reading data when BTF is set, as the data register is still full.

Bit 5 Reserved, forced by hardware to 0.

Bit 4 **STOPF**: Stop detection (slave mode)

0: No Stop condition detected
1: Stop condition detected

–Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1).

–Cleared by software reading the SR1 register followed by a write in the CR1 register, or by hardware when PE=0

Note: The STOPF bit is not set after a NACK reception.
It is recommended to perform the complete clearing sequence (READ SR1 then WRITE CR1) after the STOPF is set. Refer to Figure 236: Transfer sequence diagram for slave receiver on page 547.

Bit 3 **ADD10**: 10-bit header sent (Master mode)

0: No ADD10 event occurred.
1: Master has sent first address byte (header).

–Set by hardware when the master has sent the first byte in 10-bit address mode.

–Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by hardware when PE=0.

Note: ADD10 bit is not set after a NACK reception

Bit 2 **BTF**: Byte transfer finished

0: Data byte transfer not done
1: Data byte transfer succeeded

–Set by hardware when NOSTRETCH=0 and:

–In reception when a new byte is received (including ACK pulse) and DR has not been read yet (RxNE=1).

–In transmission when a new byte should be sent and DR has not been written yet (TxE=1).

–Cleared by software reading SR1 followed by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0.

Note: The BTF bit is not set after a NACK reception

The BTF bit is not set if next byte to be transmitted is the PEC (TRA=1 in I2C_SR2 register and PEC=1 in I2C_CR1 register)

Bit 1 **ADDR**: Address sent (master mode)/matched (slave mode)

This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0.

Address matched (Slave)

0: Address mismatched or not received.
1: Received address matched.

–Set by hardware as soon as the received slave address matched with the OAR registers content or a general call or a SMBus Device Default Address or SMBus Host or SMBus Alert is recognized. (when enabled depending on configuration).

Note: In slave mode, it is recommended to perform the complete clearing sequence (READ SR1 then READ SR2) after ADDR is set. Refer to Figure 236: Transfer sequence diagram for slave receiver on page 547.

Address sent (Master)

0: No end of address transmission
1: End of address transmission

–For 10-bit addressing, the bit is set after the ACK of the 2nd byte.

–For 7-bit addressing, the bit is set after the ACK of the byte.

Note: ADDR is not set after a NACK reception

Bit 0 **SB**: Start bit (Master mode)

0: No Start condition
1: Start condition generated.

–Set when a Start condition generated.

–Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0

22.6.7 I²C Status register 2 (I2C_SR2)

Address offset: 0x18
Reset value: 0x0000

Note: Reading I2C_SR2 after reading I2C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I2C_SR1. Consequently, I2C_SR2 must be read only when ADDR is found set in I2C_SR1 or when the STOPF bit is cleared.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

Bits 15:8 **PEC[7:0]** Packet error checking register

This register contains the internal PEC when ENPEC=1.

Bit 7 **DUALF**: Dual flag (Slave mode)

0: Received address matched with OAR1
1: Received address matched with OAR2

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 6 **SMBHOST**: SMBus host header (Slave mode)

0: No SMBus Host address
1: SMBus Host address received when SMBTYPE=1 and ENARP=1.

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 5 **SMBDEFAULT**: SMBus device default address (Slave mode)

0: No SMBus Device Default address
1: SMBus Device Default address received when ENARP=1

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 4 **GENCALL**: General call address (Slave mode)

0: No General Call
1: General Call Address received when ENGC=1

–Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 3 Reserved, forced by hardware to 0.

Bit 2 **TRA**: Transmitter/receiver

0: Data bytes received
1: Data bytes transmitted

This bit is set depending on the R/W bit of the address byte, at the end of total address phase.

It is also cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.

Bit 1 **BUSY**: Bus busy

0: No communication on the bus
1: Communication ongoing on the bus

–Set by hardware on detection of SDA or SCL low

–cleared by hardware on detection of a Stop condition.

It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).

Bit 0 **MSL**: Master/slave

0: Slave Mode
1: Master Mode

–Set by hardware as soon as the interface is in Master mode (SB=1).

–Cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.

Note: Reading I2C_SR2 after reading I2C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I2C_SR1. Consequently, I2C_SR2 must be read only when ADDR is found set in I2C_SR1 or when the STOPF bit is cleared.

22.6.8 I²C Clock control register (I2C_CCR)

Address offset: 0x1C
Reset value: 0x0000

Note: 1 f_{PCLK1} must be at least 2 MHz to achieve standard mode I²C frequencies. It must be at least 4 MHz to achieve fast mode I²C frequencies. It must be a multiple of 10MHz to reach the 400 kHz maximum I²C fast mode clock.

2 The CCR register must be configured only when the I2C is disabled (PE = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **F/S:** I2C master mode selection

0: Standard Mode I2C
1: Fast Mode I2C

Bit 14 **DUTY:** Fast mode duty cycle

0: Fast Mode t_{low}/t_{high} = 2
1: Fast Mode t_{low}/t_{high} = 16/9 (see CCR)

Bits 13:12 Reserved, forced by hardware to 0.

Bits 11:0 **CCR[11:0]:** Clock control register in Fast/Standard mode (Master mode)

Controls the SCL clock in master mode.

Standard mode or SMBus:

$T_{high} = CCR * T_{PCLK1}$
$T_{low} = CCR * T_{PCLK1}$

Fast mode:

If DUTY = 0:
$T_{high} = CCR * T_{PCLK1}$
$T_{low} = 2 * CCR * T_{PCLK1}$
If DUTY = 1: (to reach 400 kHz)
$T_{high} = 9 * CCR * T_{PCLK1}$
$T_{low} = 16 * CCR * T_{PCLK1}$
For instance: in standard mode, to generate a 100 kHz SCL frequency:
If FREQR = 08, T_{PCLK1} = 125 ns so CCR must be programmed with 0x28
(0x28 <=> 40d x 125 ns = 5000 ns.)

Note: 1. The minimum allowed value is 0x04, except in FAST DUTY mode where the minimum allowed value is 0x01
. These timings are without filters.
. The CCR register must be configured only when the I^2C is disabled (PE = 0).

22.6.9 I^2C TRISE register (I2C_TRISE)

Address offset: 0x20
Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TRISE[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 15:6 Reserved, forced by hardware to 0.

Bits 5:0 **TRISE[5:0]**: Maximum rise time in Fast/Standard mode (Master mode)

These bits must be programmed with the maximum SCL rise time given in the I^2C bus specification, incremented by 1.

For instance: in standard mode, the maximum allowed SCL rise time is 1000 ns.

If, in the I2C_CR2 register, the value of FREQ[5:0] bits is equal to 0x08 and T_{PCLK1} = 125 ns therefore the TRISE[5:0] bits must be programmed with 09h.

(1000 ns / 125 ns = 8 + 1)

The filter value can also be added to TRISE[5:0].

If the result is not an integer, TRISE[5:0] must be programmed with the integer part, in order to respect the t_{HIGH} parameter.

Note: TRISE[5:0] must be configured only when the I2C is disabled (PE = 0).

22.6.10 I²C register map

The table below provides the I²C register map and reset values.

Table 122. I²C register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**I2C_CR1**	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	**I2C_CR2**	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	**I2C_OAR1**	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	**I2C_OAR2**	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	**I2C_DR**	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	**I2C_SR1**	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	**I2C_SR2**	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	**I2C_CCR**	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**I2C_TRISE**	Reserved																										TRISE[5:0]					
	Reset value																											0	0	0	0	1	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses table.

23 Universal synchronous asynchronous receiver transmitter (USART)

23.1 USART introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It allows multiprocessor communication.

High speed data communication is possible by using the DMA for multibuffer configuration.

23.2 USART main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Fractional baud rate generator systems
  - A common programmable transmit and receive baud rates up to 4.5 MBits/s
- Programmable data word length (8 or 9 bits)
- Configurable stop bits - support for 1 or 2 stop bits
- 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
- Transmitter clock output for synchronous transmission
- IrDA SIR Encoder Decoder
  - Support for 3/16 bit duration for normal mode
- Smartcard Emulation Capability
  - The Smartcard interface supports the asynchronous protocol Smartcards as defined in ISO 7816-3 standards
  - 0.5, 1.5 Stop Bits for Smartcard operation
- Single wire half duplex communication
- Configurable multibuffer communication using DMA (direct memory access)
  - Buffering of received/transmitted bytes in reserved SRAM using centralized DMA
- Separate enable bits for Transmitter and Receiver

- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - End of Transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- Ten interrupt sources with flags:
  - CTS changes
  - LIN break detection
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Overrun error
  - Framing error
  - Noise error
  - Parity error
- Multiprocessor communication - enter into mute mode if address match does not occur
- Wake up from mute mode (by idle line detection or address mark detection)
- Two receiver wakeup modes: Address bit (MSB, 9^{th} bit), Idle line

23.3 USART functional description

The interface is externally connected to another device by three pins (see *Figure 243*). Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

RX: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

TX: Transmit Data Output. When the transmitter is disabled, the output pin returns to its IO port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and smartcard modes, this IO is used to transmit and receive the data (at USART level, data are then received on SW_RX).

Through these pins, serial data is transmitted and received in normal USART mode as frames comprising:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- 0.5,1, 1.5, 2 Stop bits indicating that the frame is complete
- This interface uses a fractional baud rate generator - with a 12-bit mantissa and 4-bit fraction
- A status register (USART_SR)
- Data register (USART_DR)
- A baud rate register (USART_BRR) - 12-bit mantissa and 4-bit fraction.
- A Guardtime Register (USART_GTPR) in case of Smartcard mode.

Refer to *Section 23.6: USART registers on page 607* for the definitions of each bit.

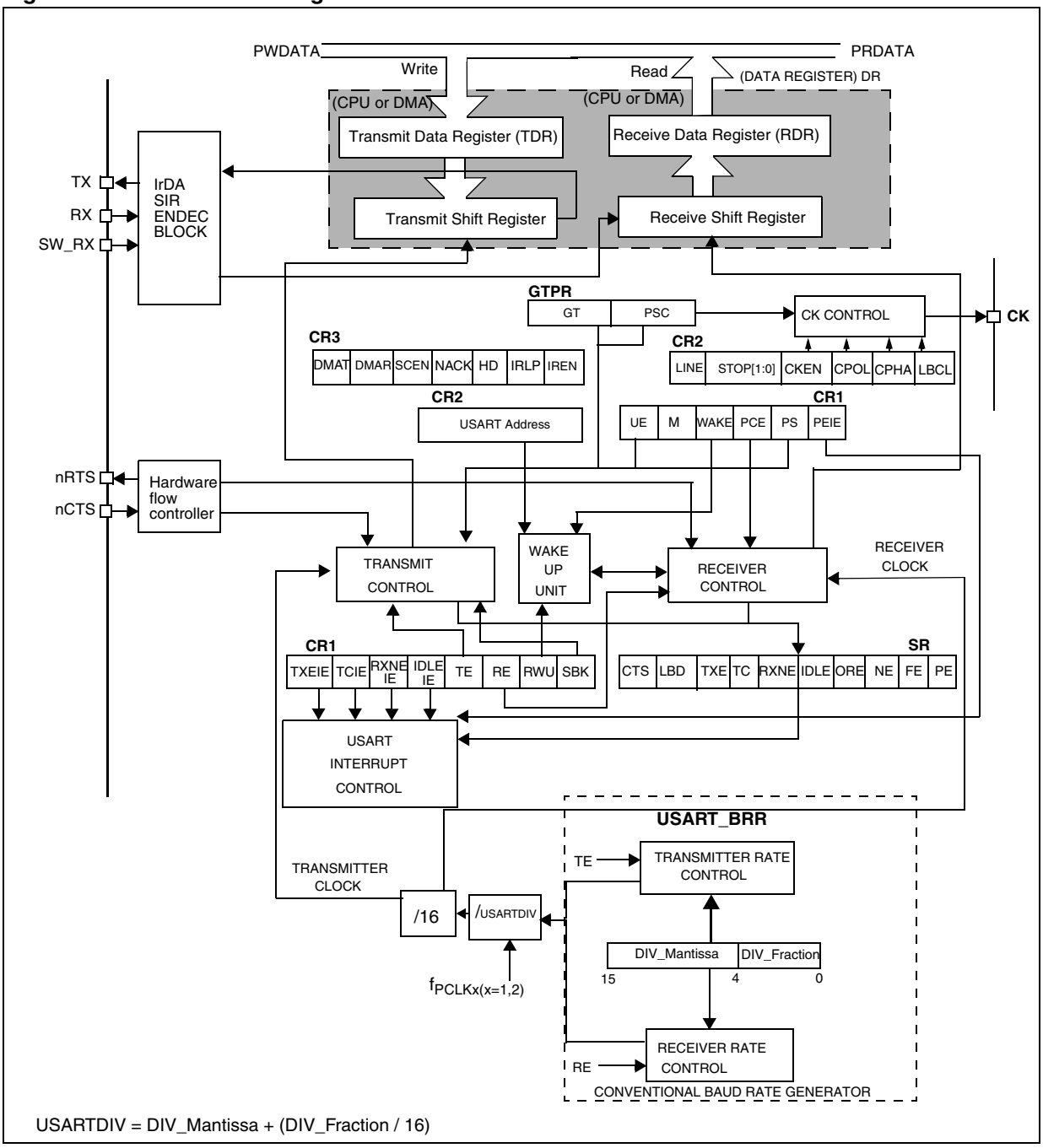
The following pin is required to interface in synchronous mode:

- **CK:** Transmitter clock output. 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. This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable. In smartcard mode, CK can provide the clock to the smartcard.

The following pins are required in Hardware flow control mode:

- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the USART is ready to receive a data (when low).

Figure 243. USART block diagram



23.3.1 USART character description

Word length may be selected as being either 8 or 9 bits by programming the M bit in the USART_CR1 register (see *Figure 244*).

The TX pin is in low state during the start bit. It is in high state during the stop bit.

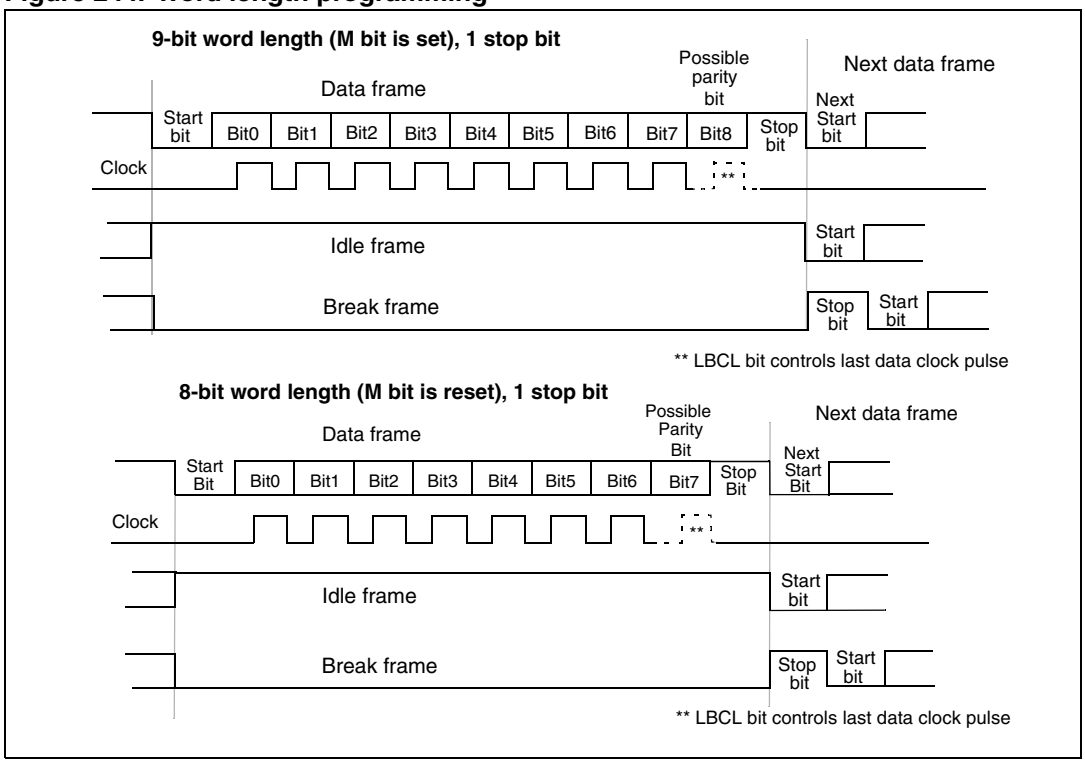
An ***Idle character*** is interpreted as an entire frame of “1”s followed by the start bit of the next frame which contains data (The number of “1” ‘s will include the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame the transmitter inserts either 1 or 2 stop bits (logic “1” bit) to acknowledge the start bit.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 244. Word length programming



23.3.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the CK pin.

Character transmission

During a USART transmission, data shifts out least significant bit first on the TX pin. In this mode, the USART_DR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 243*).

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.

Note: 1 The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.

2 An idle frame will be sent after the TE bit is enabled.

Configurable stop bits

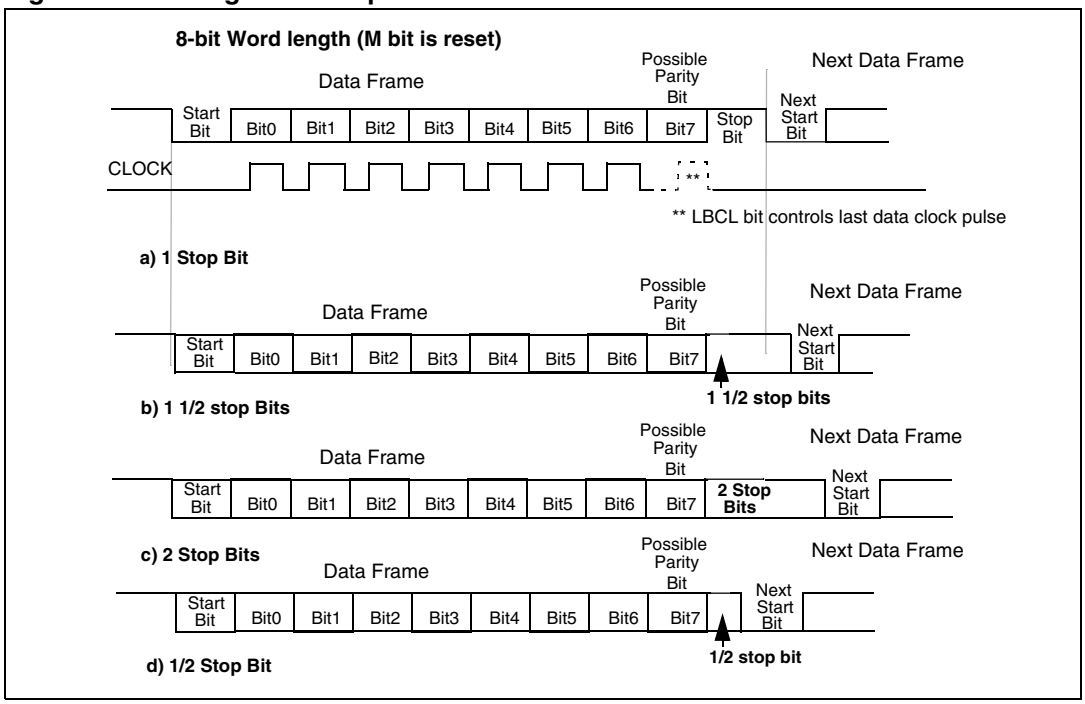
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

1. ***1 stop bit***: This is the default value of number of stop bits.
2. ***2 Stop bits***: This is supported by normal USART, single-wire and modem modes.
3. ***0.5 stop bit***: To be used when receiving data in Smartcard mode.
4. ***1.5 stop bits***: To be used when transmitting and receiving data in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits followed by the configured number of stop bits (when m = 0) and 11 low bits followed by the configured number of stop bits (when m = 1). It is not possible to transmit long breaks (break of length greater than 10/11 low bits).

Figure 245. Configurable stop bits



Procedure:

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAT) in USART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
5. Select the desired baud rate using the USART_BRR register.
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_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
8. After writing the last data into the USART_DR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

Single byte communication

The TXE bit is always cleared by a write to the data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from TDR to the shift register and the data transmission has started.
- The TDR register is empty.
- The next data can be written in the USART_DR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the USART_DR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the USART_DR register places the data directly in the shift register, the data transmission starts, and the TXE bit is immediately set.

If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

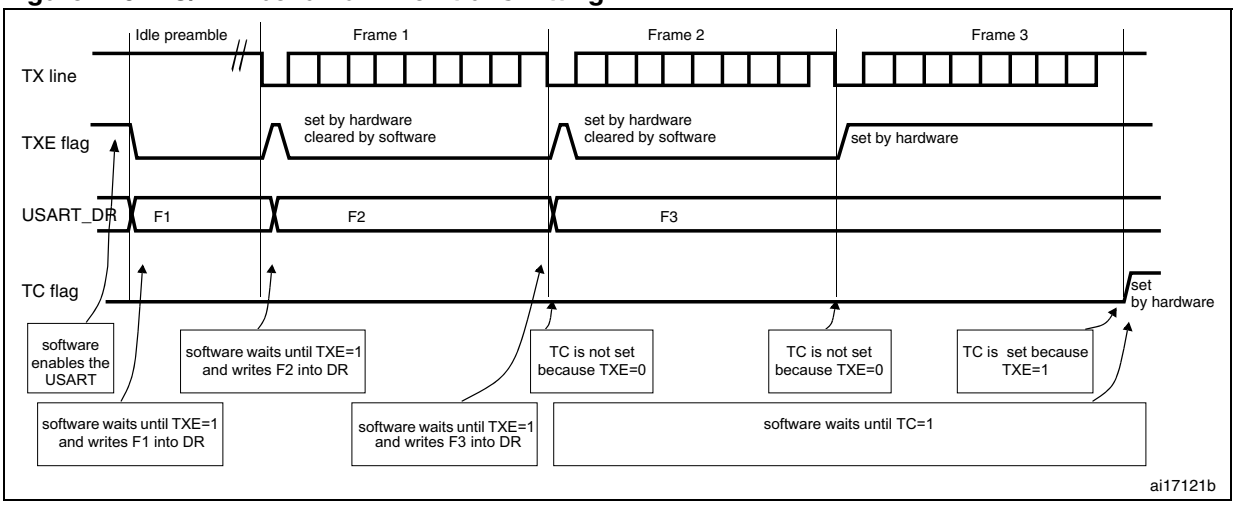
After writing the last data into the USART_DR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low power mode (see *Figure 246: TC/TXE behavior when transmitting*).

The TC bit is cleared by the following software sequence:

1. A read from the USART_SR register
2. A write to the USART_DR register

Note: The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for Multibuffer communication.

Figure 246. TC/TXE behavior when transmitting



Break characters

Setting the SBK bit transmits a break character. The break frame length depends on the M bit (see *Figure 244*).

If the SBK bit is set to '1' a break character is sent on the TX line after completing the current character transmission. This bit is reset by hardware when the break character is completed (during the stop bit of the break character). The USART inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

Note: If the software resets the SBK bit before the commencement of break transmission, the break character will not be transmitted. For two consecutive breaks, the SBK bit should be set after the stop bit of the previous break.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

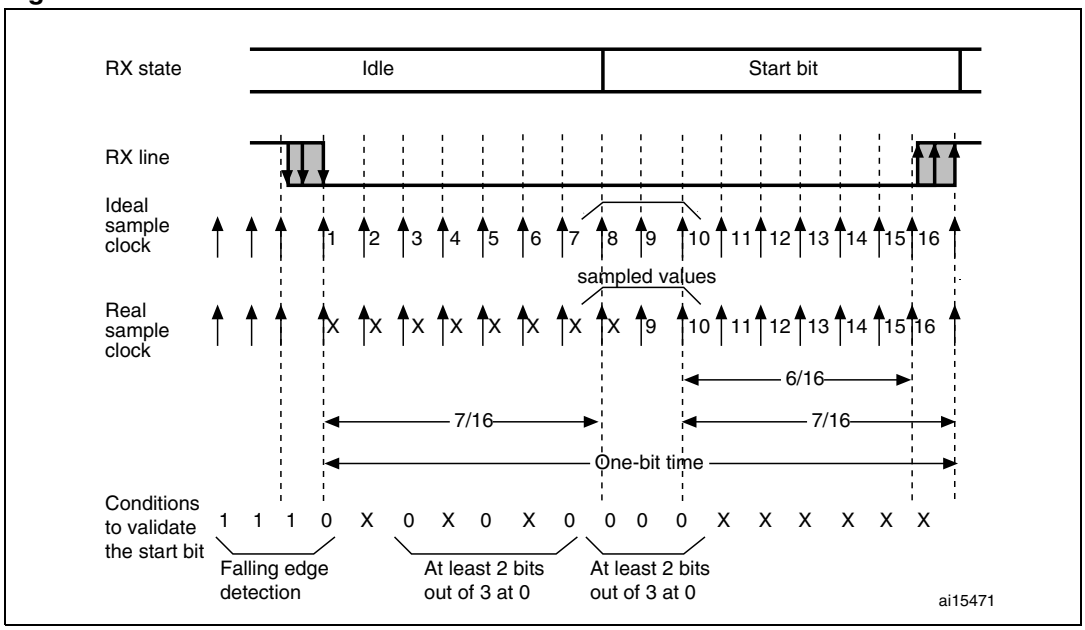
23.3.3 Receiver

The USART can receive data words of either 8 or 9 bits depending on the M bit in the USART_CR1 register.

Start bit detection

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0 X 0 0 0 0.

Figure 247. Start bit detection



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, interrupt generated if RXNEIE=1) 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 (RXNE flag set, interrupt generated if RXNEIE=1) but the NE noise flag is set if, for both samplings, at least 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). If this condition is not met, the start detection aborts and the receiver returns to the idle state (no flag is set).

If, 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, the start bit is validated but the NE noise flag bit is set.

Character reception

During a USART reception, data shifts in least significant bit first through the RX pin. In this mode, the USART_DR register consists of a buffer (RDR) between the internal bus and the received shift register.

Procedure:

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication. STEP 3
5. Select the desired baud rate using the baud rate register USART_BRR
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- 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).
- An interrupt is generated if the RXNEIE 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, RXNE is set after every byte received and is cleared by the DMA read to the Data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Note: The RE bit should not be reset while receiving data. If the RE bit is disabled during reception, the reception of the current byte will be aborted.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the IDLEIE bit is set.

Overrun error

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared.

The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- The ORE bit is set.
- The RDR content will not be lost. The previous data is available when a read to USART_DR is performed.
- The shift register will be overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXNEIE bit is set or both the EIE and DMAR bits are set.
- The ORE bit is reset by a read to the USART_SR register followed by a USART_DR register read operation.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. 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 it means that the last valid data has already been read and thus there is nothing 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. It may also occur when the new data is received during the reading sequence (between the USART_SR register read access and the USART_DR read access).*

Noise error

Over-sampling techniques are used (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise.

Figure 248. Data sampling for noise detection

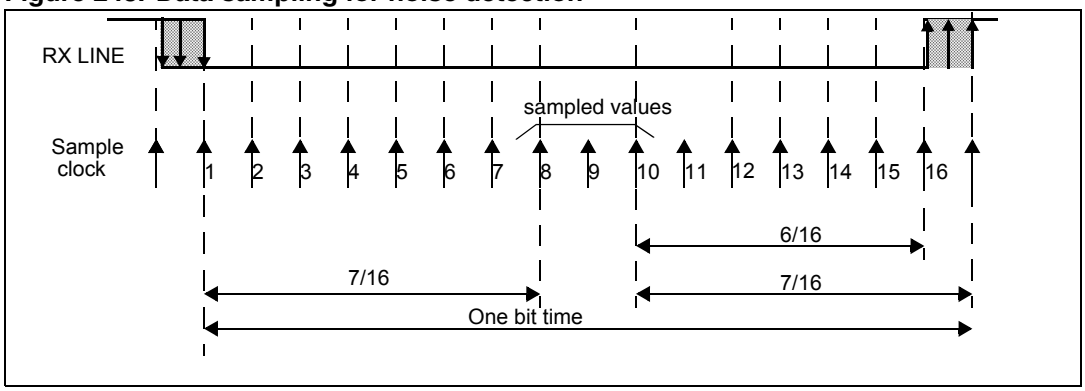


Table 123. Noise detection from sampled data

Sampled value	NE status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

When noise is detected in a frame:

- The NE is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USART_DR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.

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_DR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication an interrupt will be issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

1. ***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.
2. ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
3. ***1.5 stop bits (Smartcard mode)***: When transmitting in smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE =1 in the USART_CR1 register) 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. Then, the FE flag is set with the RXNE 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 decomposed 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 23.3.11: Smartcard on page 598* for more details.
4. ***2 stop bits***: Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag will be set. The second stop bit is not checked for framing error. The RXNE flag will be set at the end of the first stop bit.

23.3.4 Fractional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV.

$$\text{Tx/ Rx baud} = \frac{f_{CK}}{(16*\text{USARTDIV})}$$

legend: f_{CK} - Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

Note: The baud counters are updated with the new value of the Baud registers after a write to USART_BRR. Hence the Baud rate register value should not be changed during communication.

How to derive USARTDIV from USART_BRR register values

Example 1:

If DIV_Mantissa = 0d27 and DIV_Fraction = 0d12 (USART_BRR = 0x1BC), then

Mantissa (USARTDIV) = 0d27

Fraction (USARTDIV) = 12/16 = 0d0.75

Therefore USARTDIV = 0d27.75

Example 2:

To program USARTDIV = 0d25.62

This leads to:

DIV_Fraction = 16*0d0.62 = 0d9.92

The nearest real number is 0d10 = 0xA

DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19

Then, USART_BRR = 0x19A hence USARTDIV = 0d25.625

Example 3:

To program USARTDIV = 0d50.99

This leads to:

DIV_Fraction = 16*0d0.99 = 0d15.84

The nearest real number is 0d16 = 0x10 => overflow of DIV_frac[3:0] => carry must be added up to the mantissa

DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33

Then, USART_BRR = 0x330 hence USARTDIV = 0d51.000

Table 124. Error calculation for programmed baud rates

Baud rate		f_{PCLK} = 36 MHz			f_{PCLK} = 72 MHz		
S.No	in Kbps	Actual	Value programmed in the Baud Rate register	% Error =(Calculated - Desired)B.Rate /Desired B.Rate	Actual	Value programmed in the Baud Rate register	% Error
1.	2.4	2.400	937.5	0%	2.4	1875	0%
2.	9.6	9.600	234.375	0%	9.6	468.75	0%
3.	19.2	19.2	117.1875	0%	19.2	234.375	0%
4.	57.6	57.6	39.0625	0%	57.6	78.125	0.%
5.	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6.	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7.	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8.	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9.	2250	2250	1	0%	2250	2	0%
10.	4500	NA	NA	NA	4500	1	0%

Note: 1 The lower the CPU clock the lower will be the accuracy for a particular Baud rate. The upper limit of the achievable baud rate can be fixed with this data.

2 Only USART1 is clocked with PCLK2 (72 MHz Max). Other USARTs are clocked with PCLK1 (36 MHz Max).

23.3.5 USART receiver's tolerance to clock deviation

The USART's asynchronous receiver works correctly only if the total clock system deviation is smaller than the USART receiver's tolerance. 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's 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)

DTRA + DQUANT + DREC + DTCL < USART receiver's tolerance

The USART receiver's tolerance to properly receive data is equal to the maximum tolerated deviation and depends on the following choices:

- 10- or 11-bit character length defined by the M bit in the USART_CR1 register
- use of fractional baud rate or not

Table 125. USART receiver's tolerance when DIV_Fraction is 0

M bit	NF is an error	NF is don't care
0	3.75%	4.375%
1	3.41%	3.97%

Table 126. USART receiver's tolerance when DIV_Fraction is different from 0

M bit	NF is an error	NF is don't care
0	3.33%	3.88%
1	3.03%	3.53%

Note: The figures specified in Table 125 and Table 126 may slighly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M=0 (11-bit times when M=1).

23.3.6 Multiprocessor communication

There is a possibility of performing multiprocessor communication with the USART (several USARTs connected in a network). For instance, one of the USARTs can be the master, its TX output is connected to the RX input of the other USART. The others are slaves, 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 should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_CR1 register is set to 1. RWU can be controlled automatically by hardware or written by the software 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.

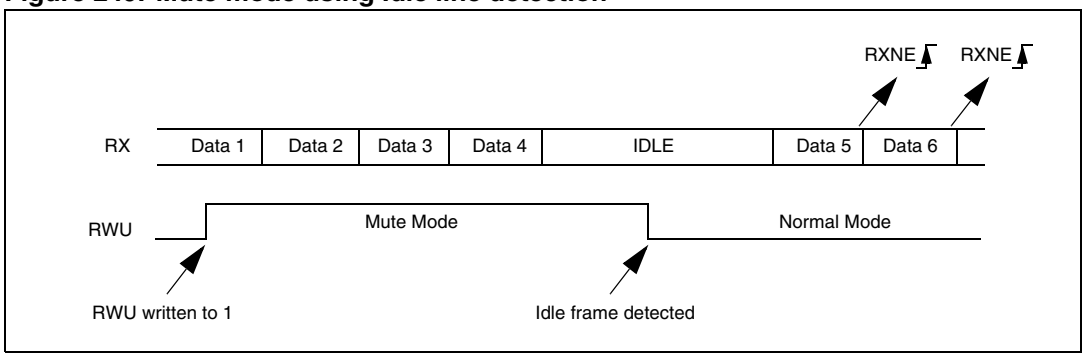
Idle line detection (WAKE=0)

The USART enters mute mode when the RWU bit is written to 1.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_SR register. RWU can also be written to 0 by software.

An example of mute mode behavior using idle line detection is given in *Figure 249*.

Figure 249. Mute mode using Idle line detection



Address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’ else they are considered as data. In an address byte, the address of the targeted receiver is put on the 4 LSB. This 4-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

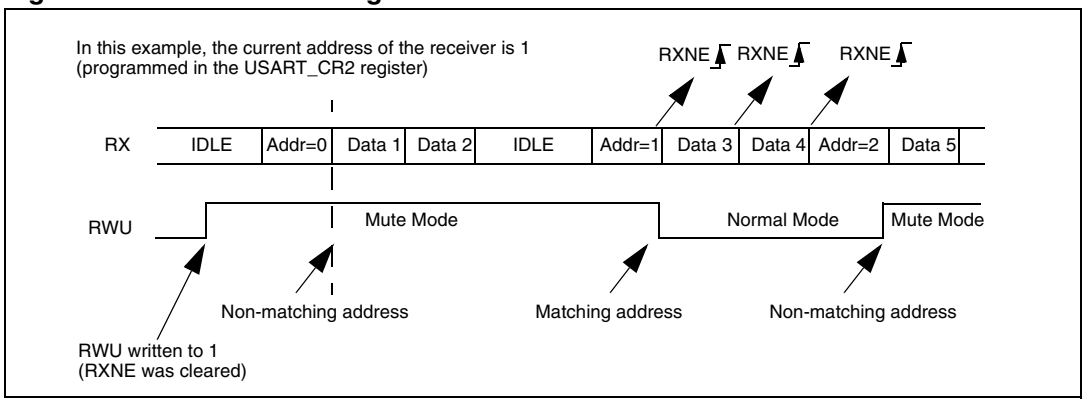
The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt nor DMA request is issued as the USART would have entered mute mode.

It 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 bit is set for the address character since the RWU bit has been cleared.

The RWU bit can be written to as 0 or 1 when the receiver buffer contains no data (RXNE=0 in the USART_SR register). Otherwise the write attempt is ignored.

An example of mute mode behavior using address mark detection is given in *Figure 250*.

Figure 250. Mute mode using address mark detection



23.3.7 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 bit, the possible USART frame formats are as listed in *Table 127*.

Table 127. Frame formats(1)

M bit	PCE bit	USART frame
0	0	ǀ SB ǀ 8 bit data ǀ STB ǀ
0	1	ǀ SB ǀ 7-bit data ǀ PB ǀ STB ǀ
1	0	ǀ SB ǀ 9-bit data ǀ STB ǀ
1	1	ǀ SB ǀ 8-bit data PB ǀ STB ǀ

1. Legends: SB: Start Bit, STB: Stop Bit, PB: Parity Bit

Note: In case of wake up by an address mark, the MSB bit of the data is taken into account and not the parity bit

Even parity: the parity bit is calculated to obtain an even number of “1s” inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Ex: data=00110101; 4 bits set => parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity: the parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Ex: data=00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Transmission mode: 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)). If the parity check fails, the PE flag is set in the USART_SR register and an interrupt is generated if PEIE is set in the USART_CR1 register.

23.3.8 LIN (local interconnection network) mode

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The same procedure explained in *Section 23.3.2* has to be applied for LIN Master transmission than 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 SBK bit sends 13 ‘0’ bits as a break character. Then a bit of value ‘1’ is sent to allow the next start detection.

LIN reception

A break detection circuit is implemented in the USART. 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 LBD flag is set in USART_SR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1’ is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0’, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1’, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 251: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 594*.

Examples of break frames are given on *Figure 252: Break detection in LIN mode vs. Framing error detection on page 595*.

Figure 251. Break detection in LIN mode (11-bit break length - LBDL bit is set)

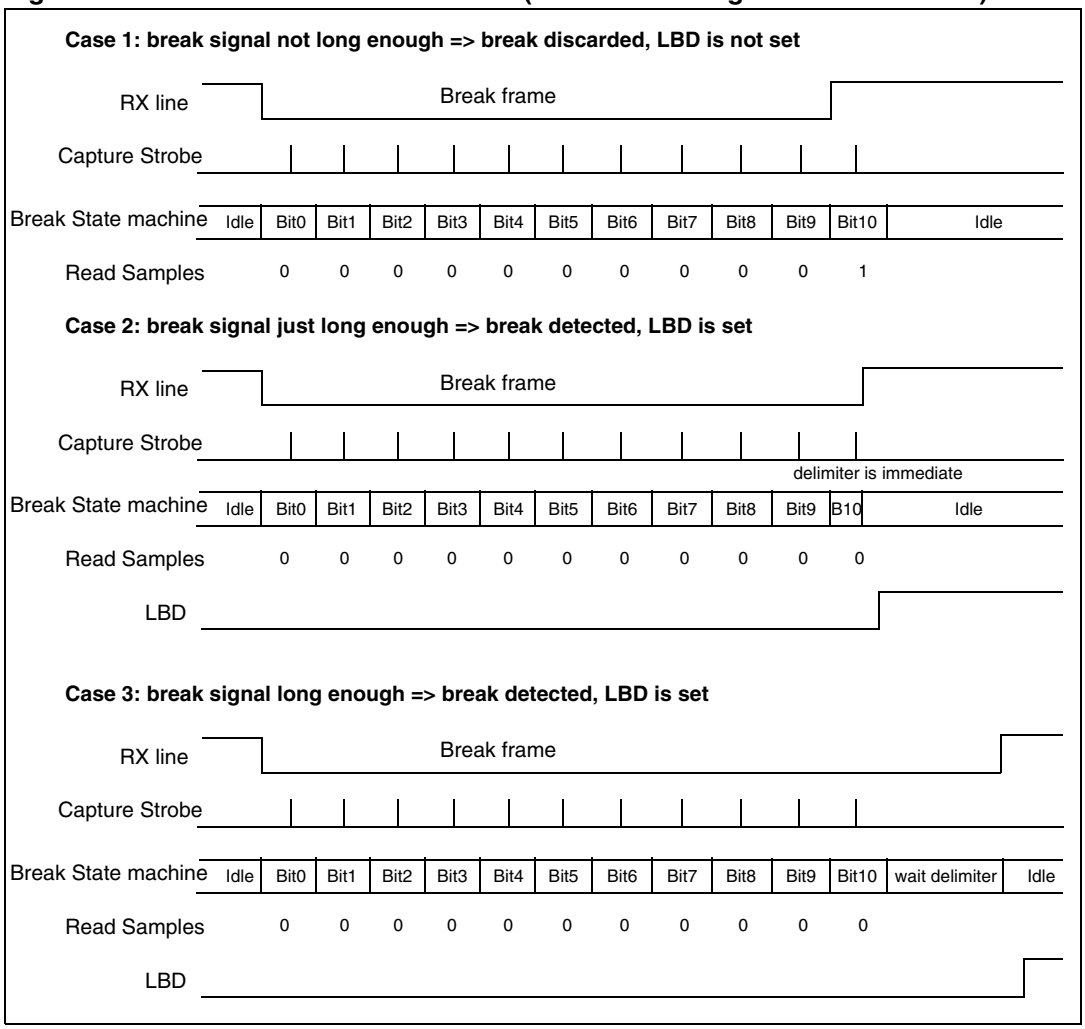
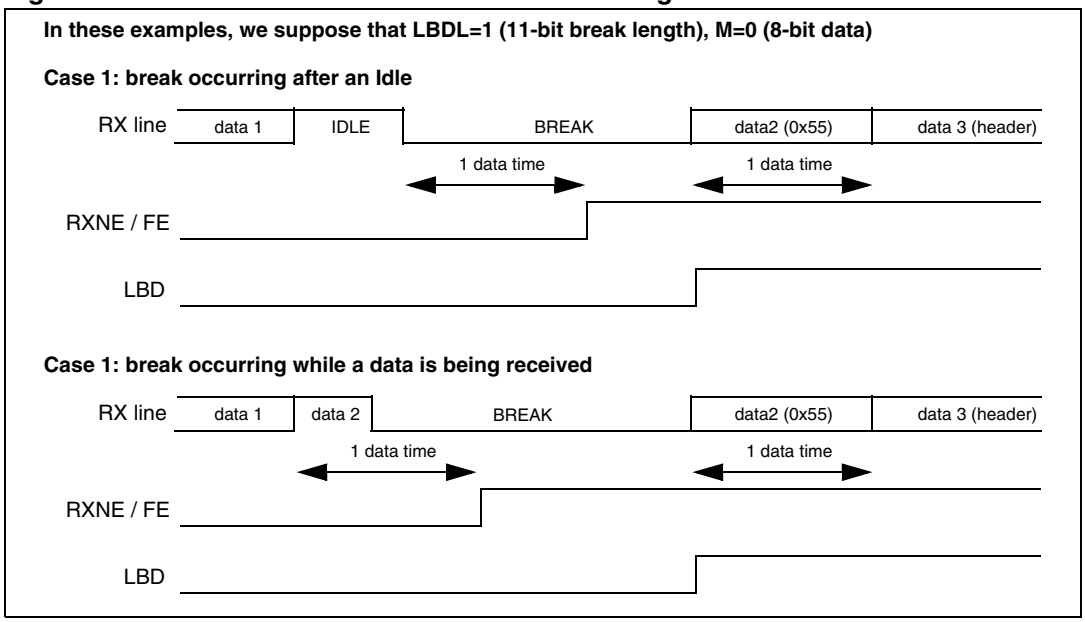


Figure 252. Break detection in LIN mode vs. Framing error detection



23.3.9 USART synchronous mode

The synchronous mode is selected by writing 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.

The USART allows the user to control a 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 will or will not be generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register allows the user to select the clock polarity, and the CPHA bit in the USART_CR2 register allows the user to select the phase of the external clock (see *Figure 253*, *Figure 254* & *Figure 255*).

During idle, preamble and send break, the external CK clock is not activated.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: 1 The CK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and a data is being transmitted (the data register USART_DR has been written). This means that it is not possible to receive a synchronous data without transmitting data.

2 The LBCL, CPOL and CPHA bits have to be selected when both the transmitter and the receiver are disabled (TE=RE=0) to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter or the receiver is enabled.

3 It is advised that TE and RE are set in the same instruction in order to minimize the setup and the hold time of the receiver.

4 The USART supports master mode only: it cannot receive or send data related to an input clock (CK is always an output).

Figure 253. USART example of synchronous transmission

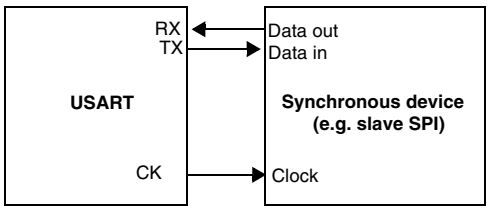


Figure 254. USART data clock timing diagram (M=0)

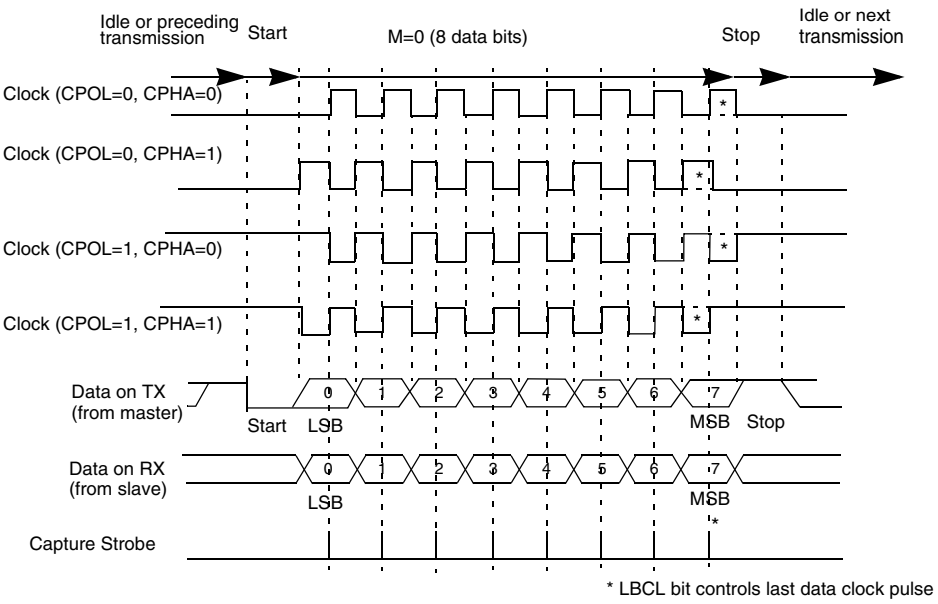


Figure 255. USART data clock timing diagram (M=1)

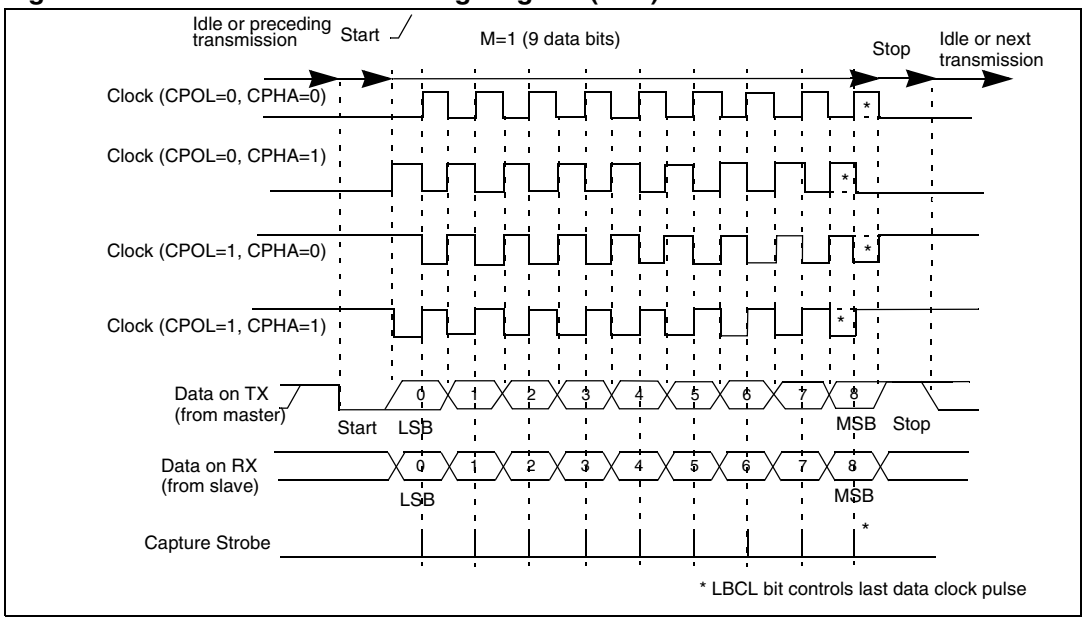
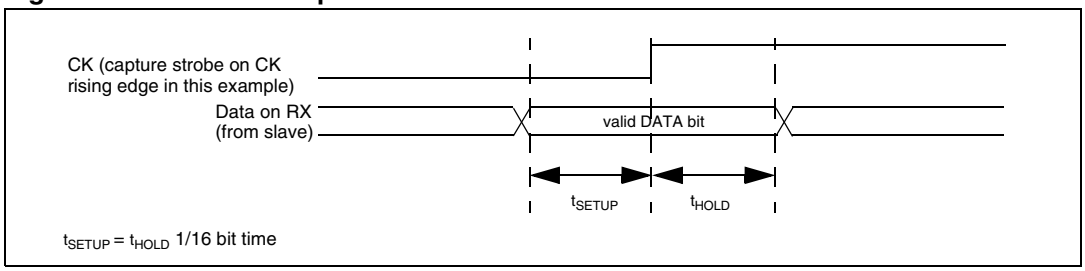


Figure 256. RX data setup/hold time



Note: The function of CK is different in Smartcard mode. Refer to the Smartcard mode chapter for more details.

23.3.10 Single-wire half-duplex communication

The 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. In single-wire half-duplex mode, the TX and RX pins are connected internally. The selection between half- and full-duplex communication is made with a control bit ‘HALF DUPLEX SEL’ (HDSEL in USART_CR3).

As soon as HDSEL is written to 1:

- RX is no longer used,
- TX is always released when no data is transmitted. Thus, it acts as a standard IO in idle or in reception. It means that the IO must be configured so that TX is configured as floating input (or output high open-drain) when not driven by the USART.

Apart from this, the communications are similar to what is done in normal USART mode. The conflicts on the line must be managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continue to occur as soon as a data is written in the data register while the TE bit is set.

23.3.11 Smartcard

The 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.

Moreover, the CLKEN bit may be set in order to provide a clock to the smartcard.

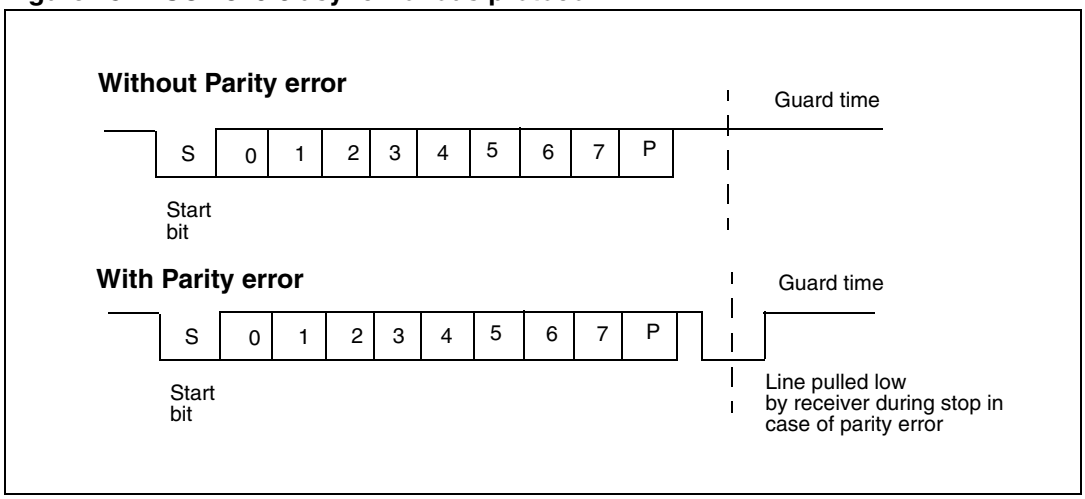
The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard. The USART should be configured as:

- 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving : where STOP='11' in the USART_CR2 register.

Note: It is also possible to choose 0.5 stop bit for receiving but it is recommended to use 1.5 stop bits for both transmitting and receiving to avoid switching between the two configurations.

Figure 257 shows examples of what can be seen on the data line with and without parity error.

Figure 257. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that the smartcard also drives into. To do so, SW_RX must be connected on the same IO than TX at product level. The Transmitter output enable TX_EN is asserted during the transmission of the start bit and the data byte, and is deasserted during the stop bit (weak pull up), so that the receive can drive the line in case of a parity error. If TX_EN is not used, TX is driven at high level during the stop bit: Thus the receiver can drive the line as long as TX is configured in open-drain.

Smartcard is a single wire half duplex communication protocol.

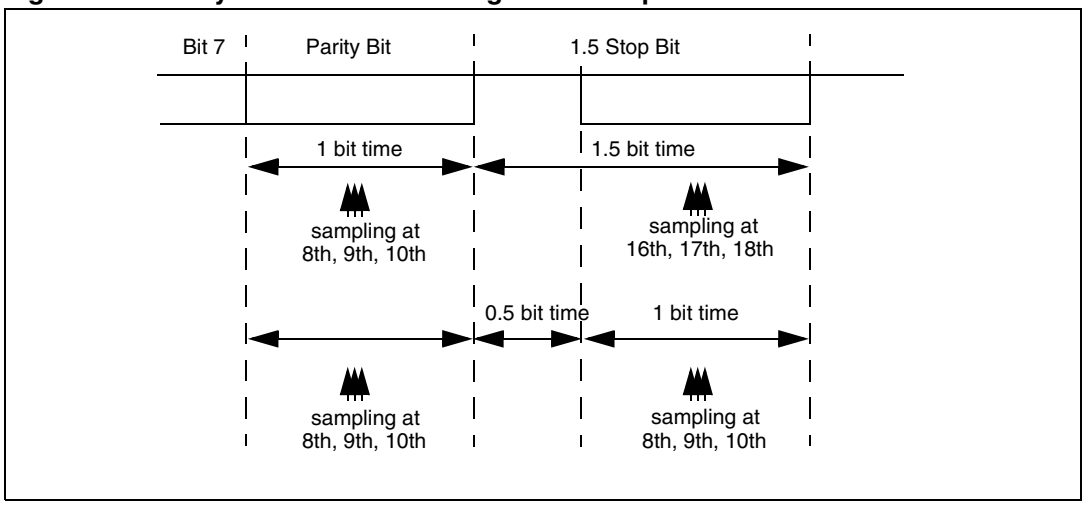
- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register will start shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- If a parity error is detected during reception of a frame programmed with a 0.5 or 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 USART has not been correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side (configured with 1.5 stop bits). The application can handle re-sending of data according to the protocol. A parity error is 'NACK'ed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted.
- 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 de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK will not be 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 will not detect the NACK as a start bit.

Note: 1 A break character is not significant in Smartcard mode. A 0x00 data with a framing error will be treated as data and not as a break.

2 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 258 details how the NACK signal is sampled by the USART. In this example, the USART transmits a 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.

Figure 258. Parity error detection using the 1.5 stop bits



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 prescaler register USART_GTPR. CK frequency can be programmed from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.

23.3.12 IrDA SIR ENDEC block

The IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 259*).

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 115.2Kbps 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 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 (i.e. the USARTsends data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (USART receives decoded data from the USART), data on the TX from the USART to IrDA is not encoded by IrDA. 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 260*).
- 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 us. 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 IrDA low-power Baud Register, USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to “1 stop bit”.

IrDA low-power mode

Transmitter:

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

Receiver:

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection, the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in USART_GTPR).

Note: 1 A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

2 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 259. IrDA SIR ENDEC- block diagram

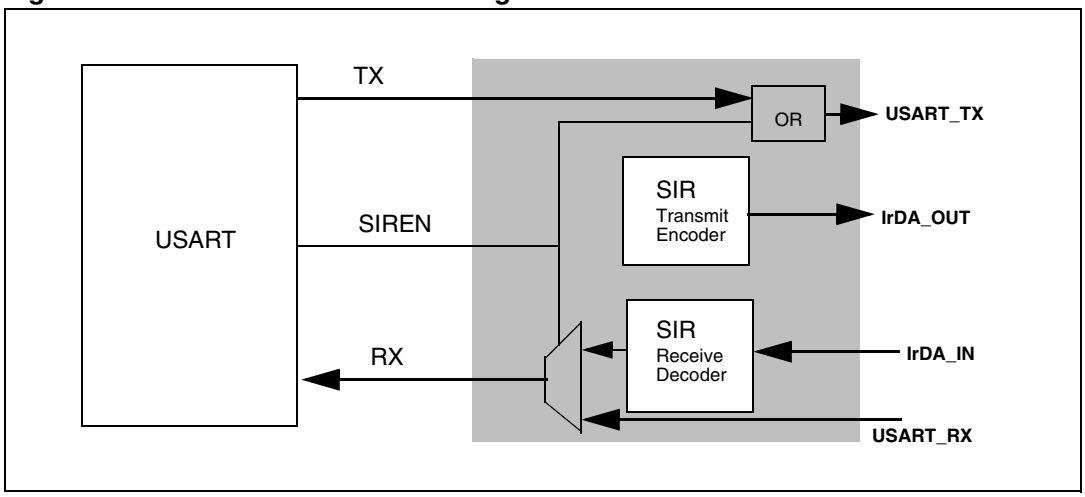
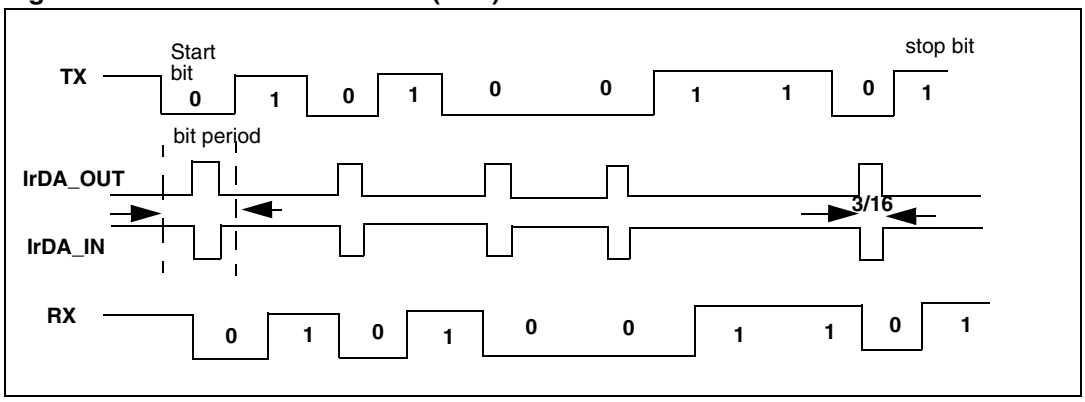


Figure 260. IrDA data modulation (3/16) -normal mode



23.3.13 Continuous communication using DMA

The USART is capable of continuing communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: You should refer to product specs for availability of the DMA controller. If DMA is not available in the product, you should use the USART as explained in Section 23.3.2 or 23.3.3. In the USART_SR register, you can clear the TXE/ RXNE flags to achieve continuous communication.

Transmission using DMA

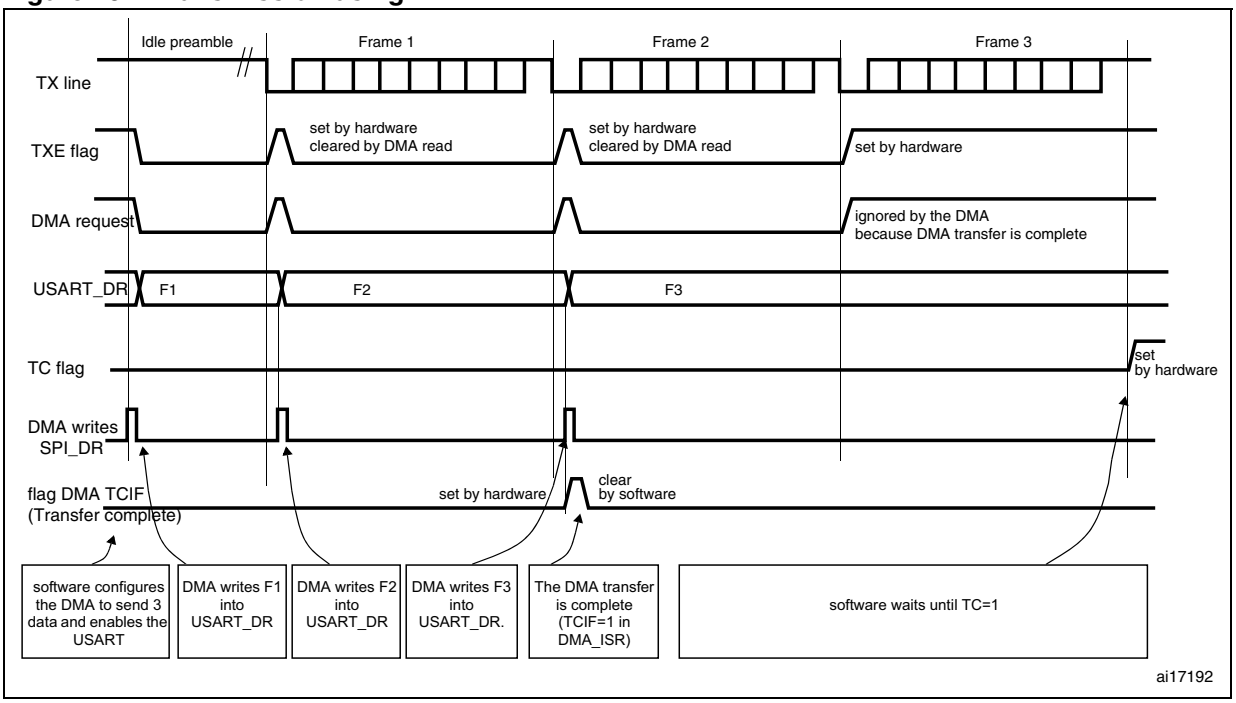
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to the DMA specification) to the USART_DR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_DR register address in the DMA control register to configure it as the destination of the transfer. The data will be moved to this address from memory after each TXE event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data will be loaded into the USART_DR register from this memory area after each TXE 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 bit in the SR register by writing 0 to it.
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 entering the Stop mode. The software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the last frame's end of transmission.

Figure 261. Transmission using DMA



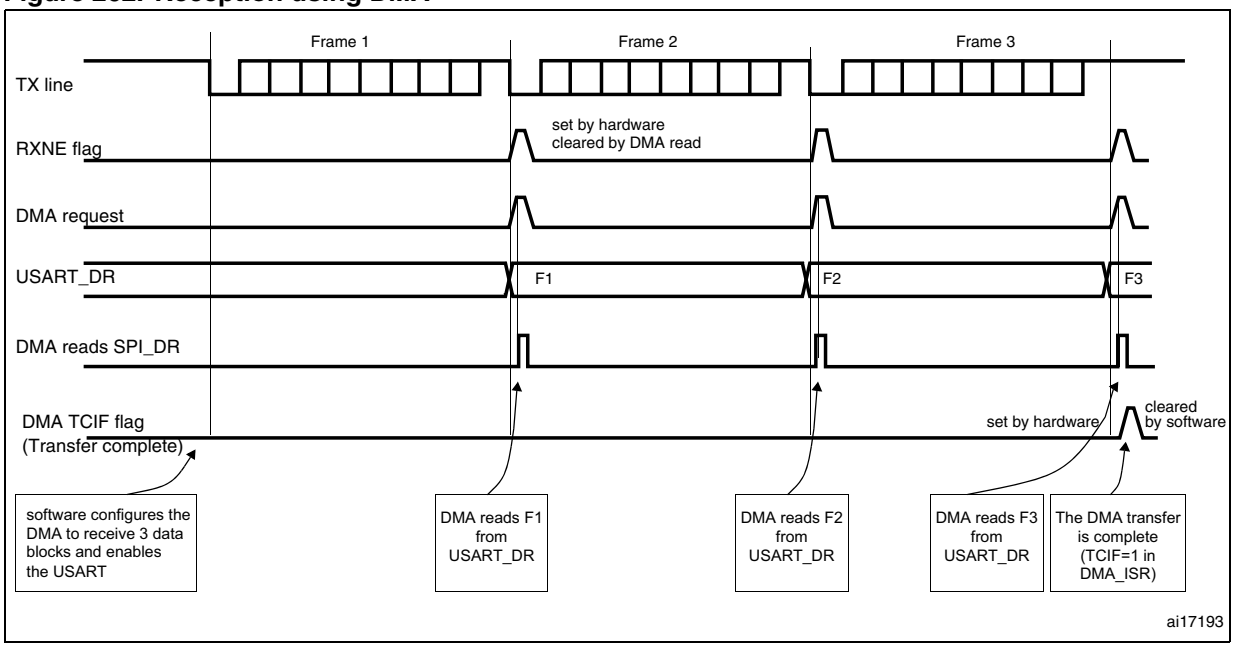
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_DR register to a SRAM area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_DR register address in the DMA control register to configure it as the source of the transfer. The data will be moved from this address to the memory after each RXNE event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data will be loaded from USART_DR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred in 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 262. Reception using DMA



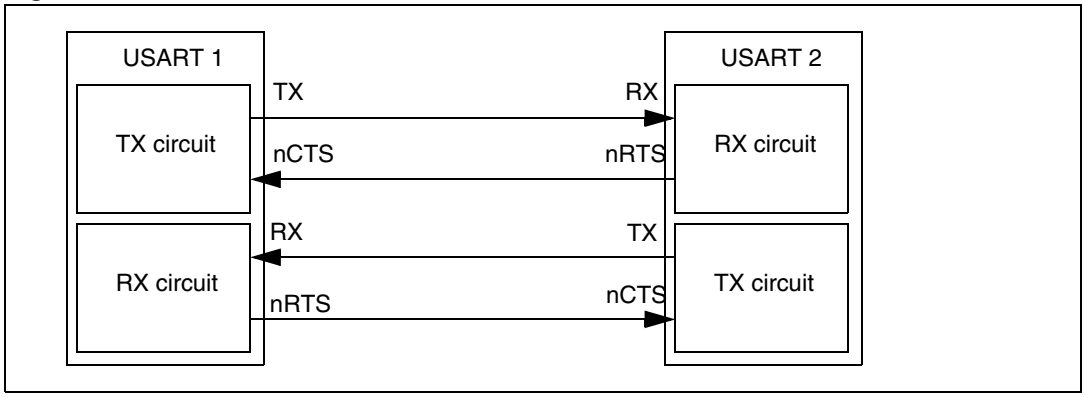
Error flagging and interrupt generation in multibuffer communication

In case of multibuffer communication if any error occurs during the transaction the error flag will be asserted after the current byte. An interrupt will be generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in case of single byte reception, there will be separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which if set will issue an interrupt after the current byte with either of these errors.

23.3.14 Hardware flow control

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. *Figure 263* shows how to connect 2 devices in this mode:

Figure 263. Hardware flow control between two USARTs

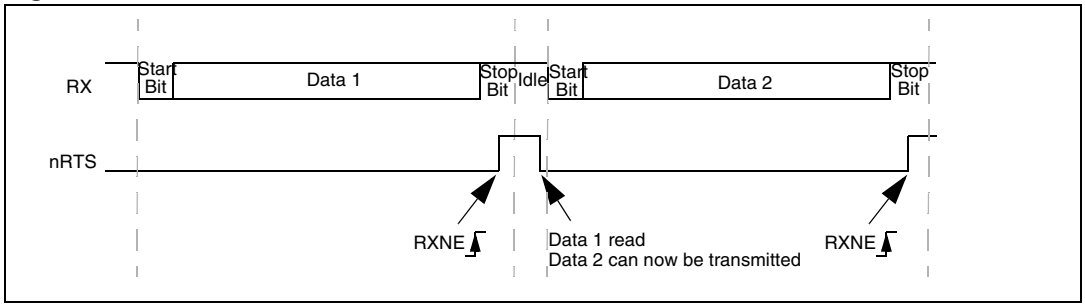


RTS and CTS flow control can be enabled independently by writing respectively RTSE and CTSE bits to 1 (in the USART_CR3 register).

RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 264* shows an example of communication with RTS flow control enabled.

Figure 264. RTS flow control

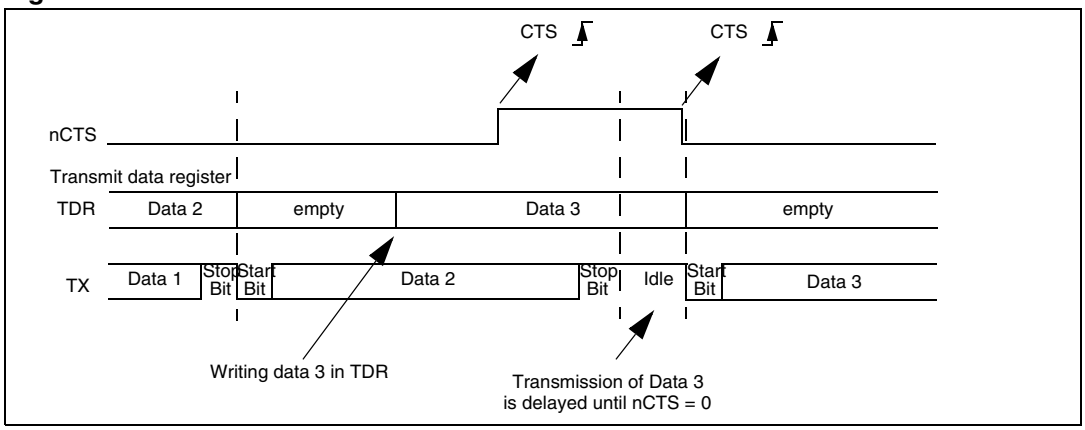


CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that a data is to be transmitted, in other words, if TXE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. The figure below shows an example of communication with CTS flow control enabled.

Figure 265. CTS flow control



23.4 USART interrupts

Table 128. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
CTS flag	CTS	CTSIE
Transmission complete	TC	TCIE
Received data ready to be read	RXNE	RXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
Break flag	LBD	LBDIE
Noise flag, Overrun error and Framing error in multibuffer communication	NE or ORE or FE	EIE(1)

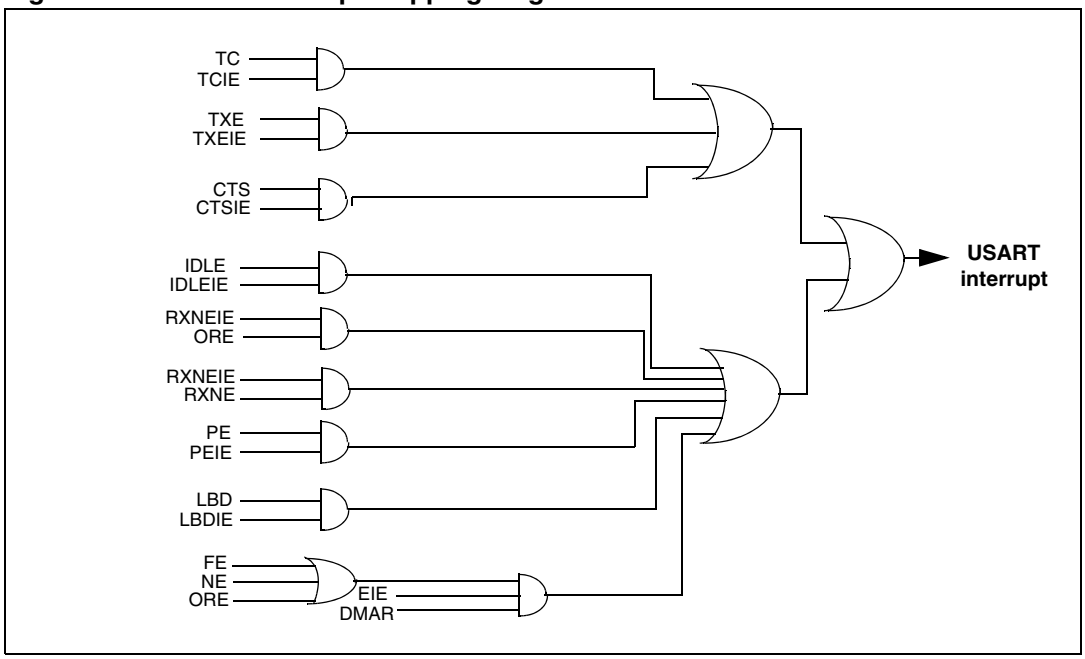
1. This bit is used only when data reception is performed by DMA.

The USART interrupt events are connected to the same interrupt vector (see *Figure 266*).

- During transmission: Transmission Complete, Clear to Send or Transmit Data Register empty interrupt.
- While receiving: Idle Line detection, Overrun error, Receive Data register not empty, Parity error, LIN break detection, Noise Flag (only in multi buffer communication) and Framing Error (only in multi buffer communication).

These events generate an interrupt if the corresponding Enable Control Bit is set.

Figure 266. USART interrupt mapping diagram



23.5 USART mode configuration

Table 129. USART mode configuration(1)

USART modes	USART1	USART2	USART3	UART4	UART5
Asynchronous mode	X	X	X	X	X
Hardware Flow Control	X	X	X	NA	NA
Multibuffer Communication (DMA)	X	X	X	X	NA
Multiprocessor Communication	X	X	X	X	X
Synchronous	X	X	X	NA	NA
Smartcard	X	X	X	NA	NA
Half-Duplex (Single-Wire mode)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

1. X = supported; NA = not applicable.

23.6 USART registers

Refer to *Section 2.1 on page 46* for a list of abbreviations used in register descriptions.

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

23.6.1 Status register (USART_SR)

Address offset: 0x00
Reset value: 0x00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Bits 31:10 Reserved, forced by hardware to 0.

Bit 9 **CTS**: CTS flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software (by writing it to 0). An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

This bit is not available for UART4 & UART5.

Bit 8 **LBD**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software (by writing it to 0). An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: An interrupt is generated when LBD=1 if LBDIE=1

Bit 7 **TXE**: Transmit data register empty

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register. It is cleared by a write to the USART_DR register.

0: Data is not transferred to the shift register

1: Data is transferred to the shift register)

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 USART_CR1 register. It is cleared by a software sequence (a read from the USART_SR register followed by a write to the USART_DR register). The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for multibuffer communication.

0: Transmission is not complete

1: Transmission is complete

Bit 5 **RXNE**: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. This clearing sequence is recommended only for multibuffer communication.

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 the IDLEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Idle Line is detected

1: Idle Line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set itself (i.e. a new idle line occurs).

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Overrun error

1: Overrun error is detected

Note: When this bit is set, the RDR register content will not be lost but the shift register will be overwritten. An interrupt is generated on ORE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 2 **NE**: Noise error flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No noise is detected

1: Noise is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupting interrupt is generated on NE flag in case of 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 a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Framing error is detected

1: Framing error or break character is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the ORE bit will be set.

An interrupt is generated on FE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read to the status register followed by a read to the USART_DR data register). The software must wait for the RXNE flag to be set before clearing the PE bit. An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

23.6.2 Data register (USART_DR)

Address offset: 0x04

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, forced by hardware to 0.

Bits 8:0 **DR[8:0]**: Data value

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR)

The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1).

The RDR register provides the parallel interface between the input shift register and the internal bus.

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.

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

23.6.3 Baud rate register (USART_BRR)

Note: The baud counters stop counting if the TE or RE bits are disabled respectively.

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, forced by hardware to 0.

Bits 15:4 **DIV_Mantissa[11:0]**: mantissa of USARTDIV

These 12 bits define the mantissa of the USART Divider (USARTDIV)

Bits 3:0 **DIV_Fraction[3:0]**: fraction of USARTDIV

These 4 bits define the fraction of the USART Divider (USARTDIV)

23.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, forced by hardware to 0.

Bit 13 **UE**: USART enable

When this bit is cleared the USART prescalers and outputs are stopped and the end of the current
byte transfer in order to reduce power consumption. This bit is set and cleared by software.
0: USART prescaler and outputs disabled
1: USART enabled

Bit 12 **M**: Word length

This bit determines the word length. It is set or cleared by software.
0: 1 Start bit, 8 Data bits, n Stop bit
1: 1 Start bit, 9 Data bits, n Stop bit

Note: The M bit must not be modified during a data transfer (both transmission and reception)

Bit 11 **WAKE**: Wakeup method

This bit determines the USART wakeup method, it is set or cleared by software.
0: Idle Line
1: Address Mark

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

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.
0: Even parity
1: Odd parity

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever PE=1 in the USART_SR register

Bit 7 **TXEIE**: TXE interrupt enable

This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever TXE=1 in the USART_SR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever TC=1 in the USART_SR register

Bit 5 **RXNEIE**: RXNE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_SR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A USART interrupt is generated whenever IDLE=1 in the USART_SR 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: 1: During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word, except in smartcard mode.
2: 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 **RWU**: Receiver wakeup
This bit determines if the USART is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wakeup sequence is recognized.
0: Receiver in active mode
1: Receiver in mute mode
Note: 1: Before selecting Mute mode (by setting the RWU bit) the USART must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.
2: In Address Mark Detection wakeup configuration (WAKE bit=1) the RWU bit cannot be modified by software while the RXNE bit is set.

Bit 0 **SBK**: Send break
This bit set is used to send break characters. It can be set and cleared by software. It should be set by software, and will be reset by hardware during the stop bit of break.
0: No break character is transmitted
1: Break character will be transmitted

23.6.5 Control register 2 (USART_CR2)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLK EN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, forced by hardware to 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 Synch Breaks (13 low bits) using the SBK bit in the USART_CR1 register, and to detect LIN Sync breaks.

Bits 13:12 **STOP**: 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 bit

The 0.5 Stop bit and 1.5 Stop bit are not available for UART4 & UART5.

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit is not available for UART4 & UART5.

Bit 10 **CPOL**: Clock polarity

This bit allows 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 is not available for UART4 & UART5.

Bit 9 **CPHA**: Clock phase

This bit allows the user 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 figures *254* to *255*)

0: The first clock transition is the first data capture edge.

1: The second clock transition is the first data capture edge.

This bit is not available for UART4 & UART5.

Bit 8 **LBCL**: Last bit clock pulse

This bit allows the user 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
The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the USART_CR1 register.
This bit is not available for UART4 & UART5.

Bit 7 Reserved, forced by hardware to 0.

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 LBD=1 in the USART_SR register

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

Bit 4 Reserved, forced by hardware to 0.

Bits 3:0 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node.
This is used in multiprocessor communication during mute mode, for wake up with address mark detection.

Note: These 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.

23.6.6 Control register 3 (USART_CR3)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, forced by hardware to 0.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited
1: An interrupt is generated whenever CTS=1 in the USART_SR register
This bit is not available for UART4 & UART5.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while a data is being transmitted, then the transmission is completed before stopping. If a data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.
This bit is not available for UART4 & UART5.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS interrupt enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (tied to 0) when a data can be received.

This bit is not available for UART4 & UART5.

Bit 7 **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

This bit is not available for UART5.

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

This bit is not available for UART5.

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bit is not available for UART4 & UART5.

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 bit is not available for UART4 & UART5.

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

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

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

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 error (FE=1 or ORE=1 or NE=1 in the USART_SR register) in case of Multi Buffer Communication (DMAR=1 in the USART_CR3 register).

0: Interrupt is inhibited

1: An interrupt is generated whenever DMAR=1 in the USART_CR3 register and FE=1 or ORE=1 or NE=1 in the USART_SR register.

23.6.7 Guard time and prescaler register (USART_GTPR)

Address offset: 0x18

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, forced by hardware to 0.

Bits 15:8 **GT[7:0]**: Guard time value

This bit-field gives the Guard time value in terms of number of baud clocks.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bit is not available for UART4 & UART5.

Bits 7:0 **PSC[7:0]**: Prescaler value

– **In IrDA Low-power mode:**

PSC[7:0] = IrDA Low-Power Baud Rate

Used for programming the prescaler for dividing the system clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

– **In normal IrDA mode:** PSC must be set to 00000001.

– **In smartcard mode:**

PSC[4:0]: Prescaler value

Used for programming the prescaler for dividing the system clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

Note: Bits [7:5] have no effect if Smartcard mode is used.

This bit is not available for UART4 & UART5.

23.6.8 USART register map

The table below gives the USART register map and reset values.

Table 130. USART register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USART_SR**	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	**USART_DR**	Reserved																							DR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x08	**USART_BRR**	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USART_CR1**	Reserved																		UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USART_CR2**	Reserved																	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																		0	0	0	0	0	0	0	0	0	0		0	0	0	0
0x14	**USART_CR3**	Reserved																					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x18	**USART_GTPR**	Reserved																GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to for the register boundary addresses.

24 High-definition multimedia interface-consumer electronics control controller (HDMI™-CEC)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xC, STM32F100xD and STM32F100xE microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to all STM32F100xx devices, unless otherwise specified.

24.1 Introduction

Consumer electronics control (CEC) is the appendix supplement 1 to the HDMI (high-definition multimedia interface) standard.

It is a protocol that provides high-level control functions between all of the various audiovisual products in an environment. It is specified to operate at low speeds with minimum processing and memory overhead.

The HDMI-CEC controller provides a hardware support of this protocol.

24.2 HDMI-CEC main features

- Supports HDMI-CEC v1.3a
- Supports the whole set of features offered with CEC (devices may use all or only some of these features, depending on functionality):
  - One touch play - a device may be played and become the active source by pressing a single button.
  - System standby - enables devices to be set on standby by pressing a single button.
  - Preset transfer - the presets of a device can be auto-configured to match those of the TV.
  - One touch record - Used to make recordings by pressing a single button.
  - Timer programming - any device can program a timer recording on a recording device.
  - System information - allows devices to auto-configure their language and country settings.
  - Deck control - allows a device to control and interrogate a playback device.
  - Tuner control - allows a device to control the tuner of another device.
  - Vendor specific commands - allows vendor-defined commands to be used.
  - OSD display - allows a device to display text using the on-screen display of the TV.
  - Device menu control - allows a device to control the menu of another device.
  - Routing control - Enables control of CEC switches for the streaming of a new source device.
  - Remote control pass through - allows remote control commands to be passed along to other devices.
  - Device OSD name transfer - devices may request the preferred OSD name of other devices within the system.

Note: If you need the power-off state, you have to use an external component/transceiver. For more details refer to AN3127: "CEC networking using STM32F100xx value line microcontrollers".

24.3 HDMI-CEC bus topology

24.3.1 HDMI-CEC pin

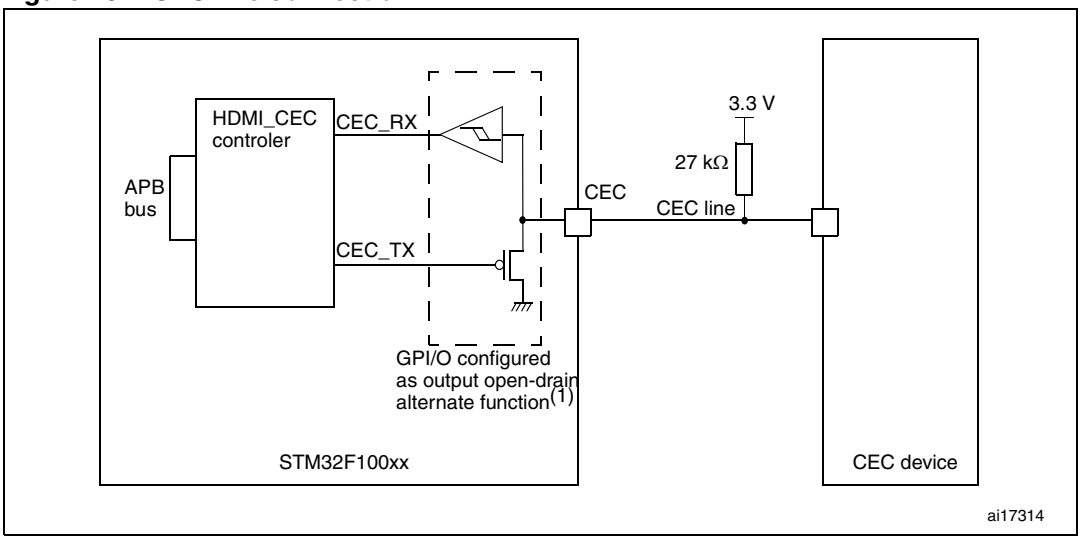
The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up resistor must be added externally to the STM32F100xx.

Table 131. HDMI pin

Name	Signal type	Remarks
CEC	Bidirectional	Two states: 1 = high impedance 0 = low impedance A 27 kΩ pull-up resistor must be added externally.

Figure 267. CEC line connection



1. When configured as output open-drain alternate function, the Schmitt trigger is still activated.

Note: 1 If you do not need the power-off state, you may simply connect the STM32F100xx microcontroller to the CEC line.

2 If the power-off state is needed in the application, the external pull-up circuit has to be disconnected from the CEC line when the device is off. For example, this can be implemented by connecting an isolating diode between the CEC line and the external pull-up circuit, such that the diode is reverse-biased in the off state with an external device pulling up the CEC line. A bidirectional isolation buffer is also needed to comply with the HDMI 1.3a specification. For more details refer to AN3127: "CEC networking using STM32F100xx value line microcontrollers".

24.3.2 Message description

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgement bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and, optionally, an opcode and a variable number of operand blocks.

All these blocks are made of a 8-bit payload (most significant bit transmitted first) followed by an end-of-message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept cleared in all others. In the event that a message contains additional blocks after an EOM is indicated, those additional blocks should be ignored. The EOM bit may be set in the header block to "ping" other devices, to ascertain if they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header or by the follower that needs to reject a broadcast message.

The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

Figure 268. Message structure

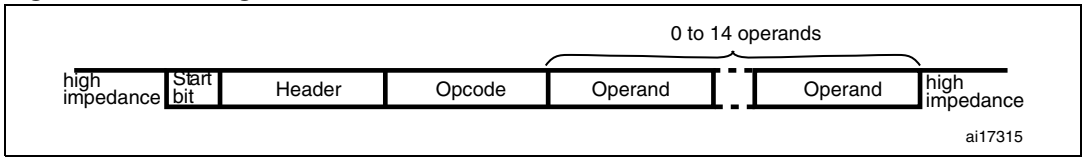
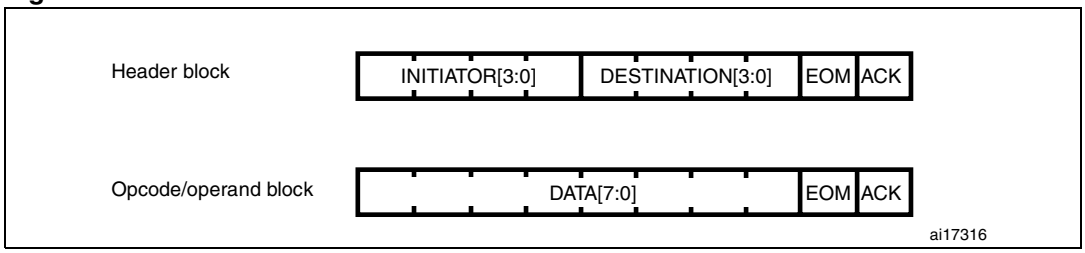


Figure 269. Blocks

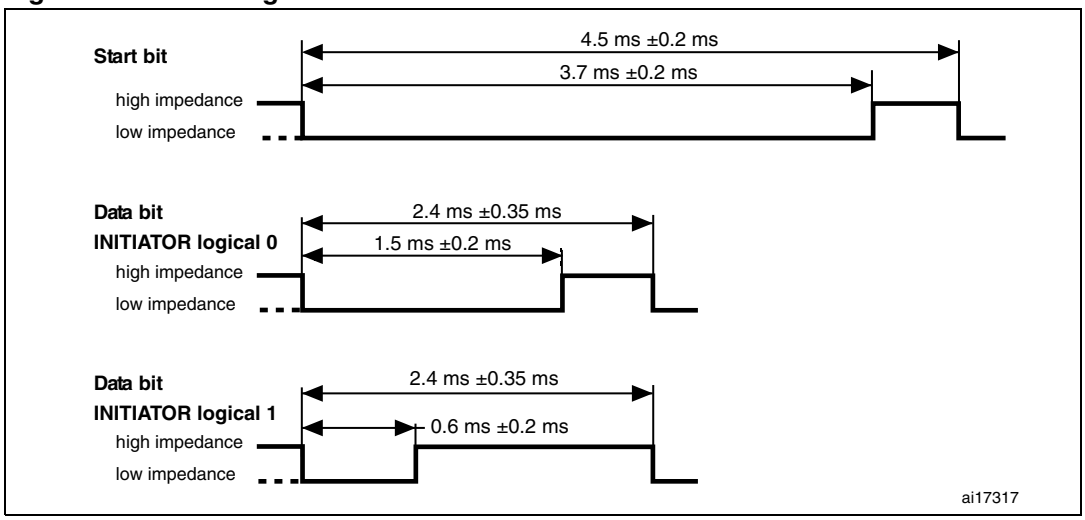


24.3.3 Bit timing

The format of the start bit is unique and identifies the start of a message. It should be validated by its low duration and its total duration.

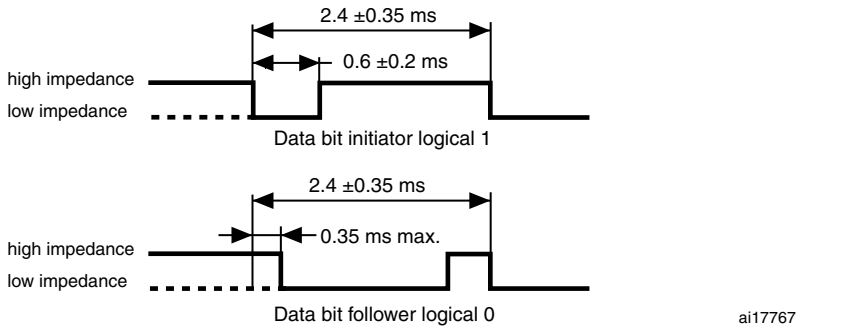
All remaining data bits in the message, after the start bit, have consistent timing. The high-to low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.

Figure 270. Bit timings



CEC *Figure 271* shows an example bit with both initiator and follower where the follower may assert the bit to logical 0 to acknowledge a data block. The initiator outputs a logical 1, thus allowing the follower to change the CEC state by pulling the control line low for the duration of the safe sample period.

Figure 271. Follower acknowledge (ACK)



24.4 Arbitration

24.4.1 Signal free time (SFT)

All devices that are to transmit or retransmit a message onto the CEC line have to ensure that the line has been inactive for a number of bit periods. This signal free time (SFT) is defined as the time from the start of the final bit of the previous frame, and depends on the initiating device and the current status as shown in the table below.

Figure 272. Signal free time

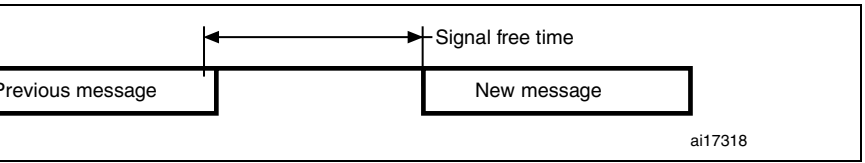
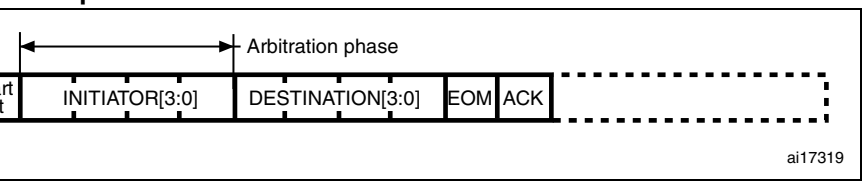


Table 132. Signal free time definition

Condition	Signal free time (in nominal data bit periods)
Present initiator has to send another frame immediately after the one it just sent	≥ 7
New initiator has to send a frame	≥ 5
Previous attempt to send frame unsuccessful	≥ 3

24.4.2 Header arbitration

Figure 273. Arbitration phase



Since only one initiator is allowed at any one time, the header arbitration mechanism is provided to avoid conflict when more than one initiator has to send a frame within the same allowed SFT slot. Header arbitration begins with the leading edge of the start bit and

continues until the end of the initiator address bits within the header block. During the header transmission period the initiator monitors the CEC line and if it detects a low impedance while it is in the high impedance state, then it assumes that it has lost the arbitration to a second initiator. Note that this process gives priority to the logical address with the highest number of leading zeros and, ultimately, the TV (INITIATOR=0x0).

24.5 Error handling

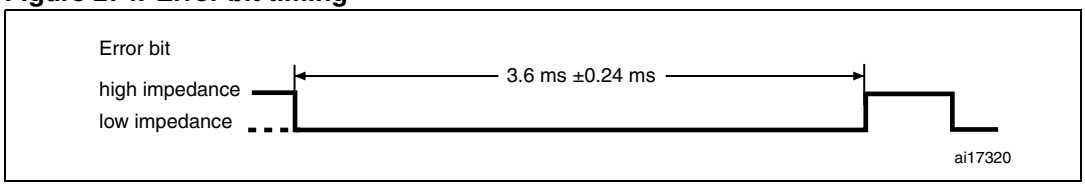
24.5.1 BTE, BPE and Error bit generation

A received data bit (excluding the start bit) is considered invalid if:

- the period between the falling and the rising edge exceeds the tolerance margins as defined by the HDMI-CEC Specification Rev1.3a, Sect5.5.2, Figure4. In this case a Bit Timing Error (BTE) is issued.
- the period between falling edges exceeds the tolerance margins as defined by the HDMI-CEC Specification, Rev1.3a Sect5.5.2, Figure4. In this case a Bit Period Error (BPE) is issued.
- When both BPE and BTE are detected, BTE only is signalled.

If a BTE or BPE receive error is detected, the CEC peripheral is expected to notify such events to the other followers, and primarily to the initiator, by generating an Error bit: a low period on the CEC line of 1.4 to 1.6 times the nominal data bit period, that is, 3.6 ms nominally.

Figure 274. Error bit timing



24.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line when not expected (line error)

Retransmission should be attempted at least once and up to five times.

24.6 Device addressing

Apart from the physical address (refer to the HDMI-CECspecification for more details on physical address discovery), each device appearing on the control signal line (CEC Line) has a unique logical address. This address defines a device type as well as being a unique identifier. This address is 0 for a TV set with physical address 0b0000 and 14 or even 15 otherwise. It is defined in the CEC_OAR register and in the upper nibble of the first byte of

the transmitted message. All CEC devices therefore have both a physical and a logical address, whereas non-CEC devices only have a physical address.

Once their physical and logical addresses are known, each CEC device transmits them to all other devices, thus allowing any device to create a map of the network.

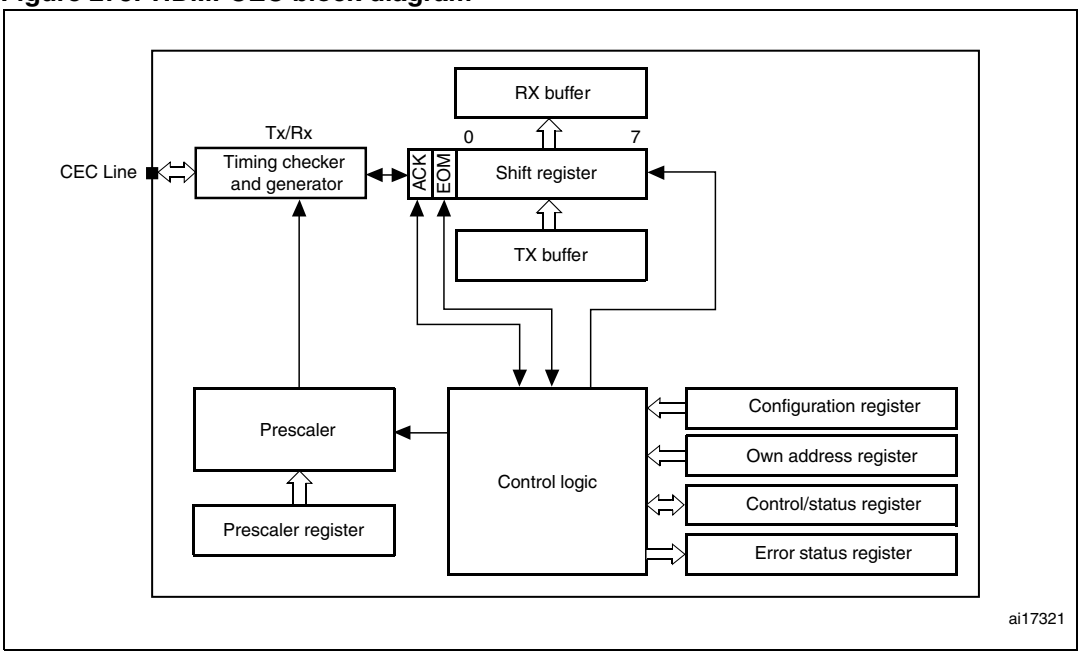
24.7 HDMI-CEC functional description

24.7.1 Block diagram

The HDMI-CEC controller handles complete messages but requires the CPU to provide or unload the data bytes one by one.

Figure 275 shows the CEC controller block diagram.

Figure 275. HDMI-CEC block diagram



1. The timing checker block verifies the received bit timings, while the timing generator controls the transmitted bit timings.

24.7.2 Prescaler

The prescaler defines the time quantum for the timing checker and timing generator blocks. Additionally, it provides a time quantum reference for complying with the required signal free time (SFT). A 14-bit counter is used to provide the necessary 50 microsecond time base, allowing high APB clocks frequency.

The counter is reset at the beginning of every bit for the timing checker block to operate with the maximum precision.

Prescaler formula for nominal bit timings is:

$$PRESC = [50 \times F_{APB}\langle MHz\rangle] - 1$$

24.7.3 Rx digital filter

CEC robustness in the face of CEC line perturbation is guaranteed by two noise rejection mechanisms:

- high-frequency spikes are removed by a 2/3 majority voter applied on the Rx line sampled at the system clock rate
- line rebounds are filtered until the next 50 µs time window entirely following any CEC line transition

24.7.4 Rx bit timing

The CEC operates at a 50 µs time quantum since the bits timings are expressed with this precision. It extracts valid bits from the CEC line and signals line errors when detected.

On a valid Rx falling edge, the value of the time counter is captured and reset. This value indicates the total bit duration and is named *m*. If the captured value is outside the valid range (see table below), a bit period error is detected and signalled by pulling the line low for 3.6 ms (typical).

On a valid Rx rising edge, the value of the time counter is captured and compared to valid windows. This value indicates the low bit duration and is named *n.* If the low bit duration (the timing between the falling and rising edges) is outside the valid range, a bit timing error is detected and signalled by pulling the line low for 3.6 ms (typical) unless the device was programmed not to report this type of violations.

Note: If a line error occurs while a start bit is expected, the whole message is ignored and no error is reported.

In the absence of a rising edge, the time counter counts up to 25.6 ms. Retransmission is allowed when its value is above 9.6 ms. A new initiator may transmit when the time counter is above 14.4 ms. The same initiator must however wait until the counter reaches 19.2 ms.

Note: Due to Rx synchronization and digital filtering, all CEC timings are calculated with a precision tolerance of 4 APB clock cycles.

Figure 276. Bit timing

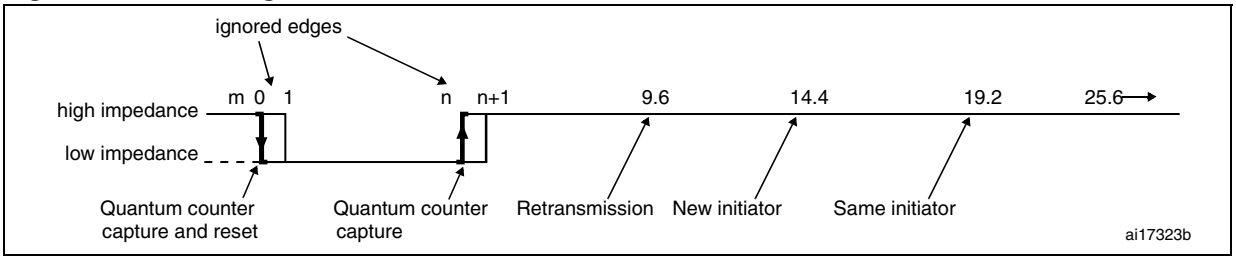


Table 133. Bit status depending on the low bit duration (LBD)

Low bit duration (LBD) (ms)	Bit	
	Standard mode (BTEM = 0)(1)	Bit timing error-free (BTEM = 1)(1)
0 ≤ LBD < 0.4	Bit timing error	Logical 1
0.4 ≤ LBD ≤ 0.8	Logical 1	
0.8 < LBD ≤ 1.1	Bit timing error	
1.1 < LBD < 1.3		Logical 0
1.3 ≤ LBD ≤ 1.7	Logical 0	
1.7 < LBD < 3.5	Bit timing error	
3.5 ≤ LBD ≤ 3.9	Start bit	Start bit
3.9 < LBD	Bit timing error	

1. BTEM is a bit of the CEC_CFGR register.

Table 134. Bit status depending on the total bit duration (TBD)

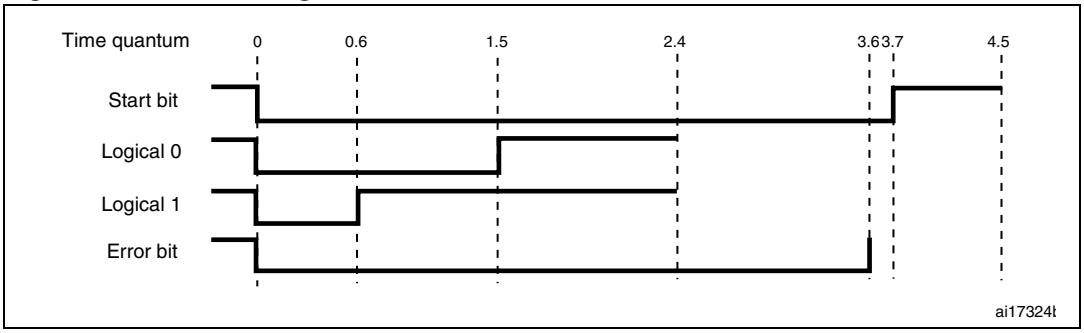
Total bit duration (TBD) (ms)	Bit		
	Standard mode (BPEM = 0)(1)	Flexible bit-period mode (BPEM = 1)(1)	
		Data bit	Start bit
TBD < 2.05	Bit period error	Bit period error	Bit period error
2.05 ≤ TBD ≤ 2.75	Logical 0 or 1	Logical 0 or 1	
2.75 < TBD < 4.3	Bit period error		
4.3 ≤ TBD ≤ 4.7	Start bit		Start bit
4.7 < TBD(2)	Bit period error		

1. BPEM is a bit of the CEC_CFGR register.
2. The bit period error checking is not applicable for the last bit in the frame.

24.7.5 Tx bit timing

The CEC is in charge of generating the proper line waveform to signal either a start bit, a logical 0 data bit, a logical 1 data bit or an error bit. The same time quantum as in the Rx bit timing logic is used.

Figure 277. Tx bit timing



24.7.6 CEC arbiter

The STM32F100xx CEC arbiter declines SFT and header arbitration techniques in the following cases:

- CEC is a previous initiator retrying a failed transmission and
  a) the leading edge of the start bit is detected before SFT = 9.6 ms. This only occurs when a new initiator violates the SFT requirement. In this case CEC automatically synchronizes to the end of the transmitted start bit and participates in the header arbitration.
  b) CEC initiates frame transmission after 4 nominal bit times (9.6 ms) of the signal free time condition. CEC automatically switches to follower in case arbitration is lost (even though no contending device is expected in this case).
- CEC is a new initiator that needs to send a frame and
  a) the leading edge of the start bit is detected before SFT = 5 minimum bit times (10.3 ms). This typically occurs when the previous initiator retries a failed transmission. In this case, CEC automatically switches to reception. Transmission is retried after the current frame.
  b) the leading edge of the start bit is detected when SFT is between 10.3 ms and 14.4 ms. This is usually caused by a different initiator contending the CEC line. STM32 CEC automatically synchronizes to the end of the transmitted start bit and participates in the header arbitration.
  c) CEC initiates frame transmission after 6 nominal bit times (14.4 ms) of the signal free time condition. It automatically switches to follower in case the arbitration is lost.
- CEC is the last initiator that has to send a frame immediately after its previous frame and
  a) the leading edge of the start bit is detected before SFT = 7 minimum bit times (14.4 ms). This is caused by a new initiator. CEC automatically switches to reception and transmission is retried after the current frame, when CEC is no longer the last initiator.
  b) the leading edge of the start bit is detected when SFT is between 14.4 ms and 19.2 ms. This might be caused by a new late initiator. CEC automatically synchronizes to the end of the transmitted start bit and participates in the header arbitration.
  c) CEC initiates frame transmission after 8 nominal bit times (19.2 ms) of the signal free time condition. CEC automatically switches to follower if the arbitration is lost (even though no contending device is expected in this case).

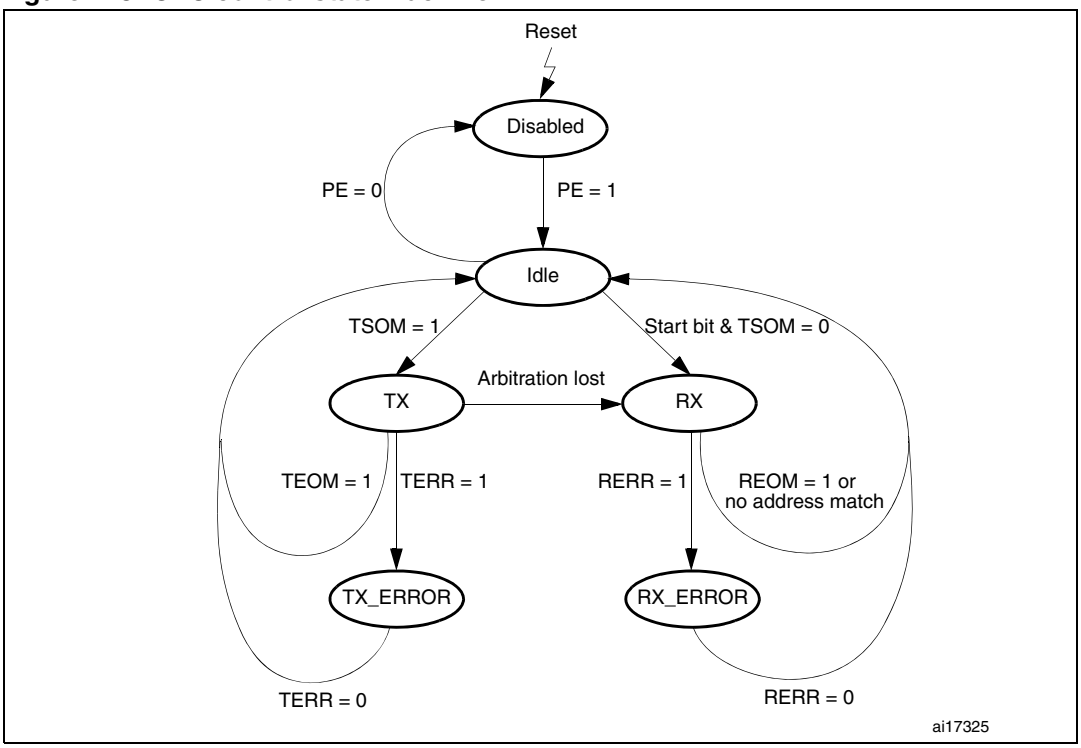
Table 135. STM32 CEC arbitration

SFT (ms)	< 9.6	< 10.3	< 14.4	< 19.2	≥ 19.2
Same initiator retrying failed transmission	enter arbitration	frame started			
New initiator	switch to reception		enter arbitration	frame started	
Same initiator has to send another frame	switch to reception			enter arbitration	frame started

24.7.7 CEC states

Figure 278 shows the CEC controller state machine.

Figure 278. CEC control state machine



The CEC controller assumes one of the six states described below:

Disabled state

The Disabled state is entered either on an APB reset or on resetting the PE bit in the CEC configuration register. Any ongoing transmission or reception is not interrupted and completes normally. The controller actually switches to Disabled when the PE bit is read back as 0. While the controller is in the Disabled state, activity on the CEC line is ignored and the clock prescaler is stopped for minimum power consumption purposes. The controller exits the Disabled state when the PE bit is set.

Idle state

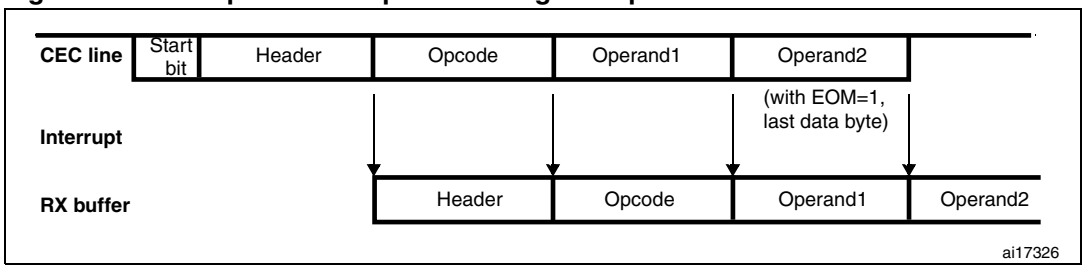
The Idle state is entered whenever a message was transmitted or received successfully, or an error was processed. While in the Idle state, the CEC controller waits for either a transmit request (TSOM bit is set in the control status register) or a start bit.

RX state

The CEC controller enters the RX state when a start bit is detected and no message is pending for transmission. Once the header has been received, the destination address is compared to the value programmed in the own address register. If the two do not match and the address is not the broadcast address 0xF, the block is not acknowledged and the controller reverts to the Idle state. Otherwise, in the case of a match, the controller remains in the RX state where the host CPU is requested to retrieve all message bytes from the RX buffer one by one. An available byte is signalled by the RBTF bit being set in the control

status register. The host CPU can either poll this register or enable interrupts in the configuration status register to know whether a byte was received. If the RBTF bit is not cleared by the time a new block is received, the newly received block is not acknowledged to force the initiator to restart the message transmission, thus giving the host CPU a second chance to retrieve all message bytes in time. Note that it is the responsibility of the software driver to ignore messages where the number of operands is less than the number specified for the opcode.
The figure below shows an example of a complete message reception.

Figure 279. Example of a complete message reception



The software has to respect the sequence described in the table below.

Table 136. Software sequence to respect when receiving a message

Software sequences	CEC_CSR register R/W access	Status bits			
		RTBF bit 7	RERR bit 6	REOM bit 5	RSOM bit 4
Poll RBTF or wait until an interrupt occurs	Read 0x00	0	0	0	0
A header is received (RTBF and RSOM are set)	Read 0x90 or 0x91(1)	1	0	0	1
Read header from RX buffer	-	1	0	0	1
Acknowledge received byte by writing 0x00	Write 0x00	0	0	0	0
Poll RBTF or wait until an interrupt occurs	Read 0x00	0	0	0	0
An opcode is received (RBTF is set)	Read 0x80 or 0x81	1	0	0	0
Read opcode from RX buffer	-	1	0	0	0
Acknowledge received byte by writing 0x00	Write 0x00	0	0	0	0
Poll RBTF or wait until an interrupt occurs	Read 0x00	0	0	0	0
An operand is received (RBTF is set)	Read 0x80 or 0x81	1	0	0	0
Read Operand1 from RX buffer	-	1	0	0	0
Acknowledge received byte by writing 0x00	Write 0x00	0	0	0	0
Poll RBTF or wait until an interrupt occurs	Read 0x00	0	0	0	0
An operand is received, which is the last data byte (RBTF and REOM are set)	Read 0xA0 or 0xA1	1	0	1	0
Read Operand2 from RX buffer	-	1	0	1	0
Acknowledge received byte by writing 0x00	Write 0x00	0	0	0	0

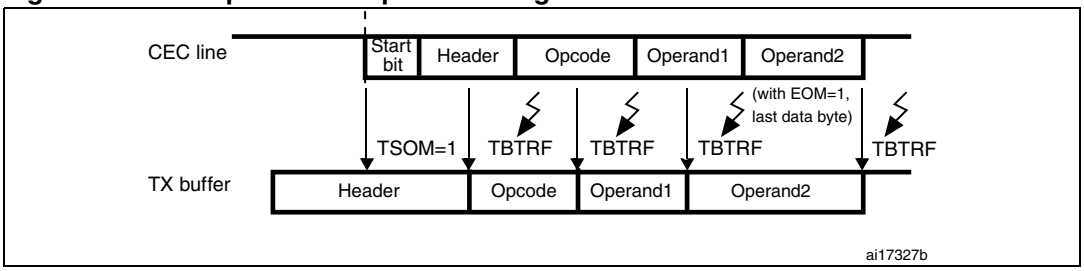
1. Two different values may be read from the control and status register since a message may have queued for transmission but arbitration has been lost.

TX state

The controller enters the TX state when the TSOM bit is set in the control status register. In this state, it has to make sure that the required signal free time elapses before generating a start bit. That is, it has to wait for 9.6 ms if the previous state was TX_ERROR, 14.4 ms if the device was previously receiving, 19.2 ms otherwise. This wait count is however abandoned if another device transmits a start bit. At this point the CEC arbiter decides whether to switch to reception or to participate in the arbitration phase. Note that it is the responsibility of the software driver to send an initiator address consistent with the logical address programmed in the own address register.
Arbitration is lost if the received initiator address, contained in the least significant nibble of the shift register, differs from the initiator address still present in the TX buffer. In this case, the controller switches to the RX state immediately. After the receive phase, it however automatically retries transmitting until it is granted ownership of the bus.
If arbitration is not lost, a new byte should be written to the TX buffer each time the TBTRF bit is set in the control status register. The host CPU can either poll the control register or enable interrupts in the configuration register, to know whether a byte was transmitted. If it does not achieve the required task on time, a transmit error flag is set. The TEOM bit is set in the control status register to indicate that the message transmission was successful, but if the TERR bit is set in the control status register, the message should be considered lost.

The following shows an example of a complete message transmission.

Figure 280. Example of a complete message transmission



The software must respect the sequence described below.

Table 137. Software sequence to respect when transmitting a message

Software sequences	CEC_CSR R/W access	Status bits			
		TBTRF bit 3	TERR bit 2	TEOM bit 1	TSOM bit 0
Write header to TX buffer	-	0	0	0	0
Initiate message transmission by writing TSOM=1	Write 0x01	0	0	0	1
Poll TBTRF or wait until an interrupt occurs	Read 0x01/0x00	0	0	0	1 then 0
The TX buffer is empty (TBTRF is set)	Read 0x08	1	0	0	0
Write opcode to TX buffer	-	1	0	0	0
Acknowledge byte request by writing 0x00	Write 0x00	0	0	0	0
Poll TBTRF or wait until an interrupt occurs	Read 0x00	0	0	0	0
The TX buffer is empty (TBTRF is set)	Read 0x08	1	0	0	0

Table 137. Software sequence to respect when transmitting a message (continued)

Software sequences	CEC_CSR R/W access	Status bits			
		TBTRF bit 3	TERR bit 2	TEOM bit 1	TSOM bit 0
Write Operand1 to TX buffer	-	1	0	0	0
Acknowledge byte request by writing 0x00	Write 0x00	0	0	0	0
Poll TBTRF or wait until an interrupt occurs	Read 0x00	0	0	0	0
Write Operand2 to TX buffer	-	1	0	0	0
Acknowledge byte request and signal end of message	Write 0x02	0	0	1	0
Poll TBTRF or wait until an interrupt occurs	Read 0x00	0	0	0	0
Message transmission is completed (TBTRF and TEOM are set)	Read 0x0A	1	0	1	0
Acknowledge successful completion by writing 0x0	Write 0x0	0	0	0	0

TX_ERROR state

The TX_ERROR state is entered if one of the following error conditions occurs:

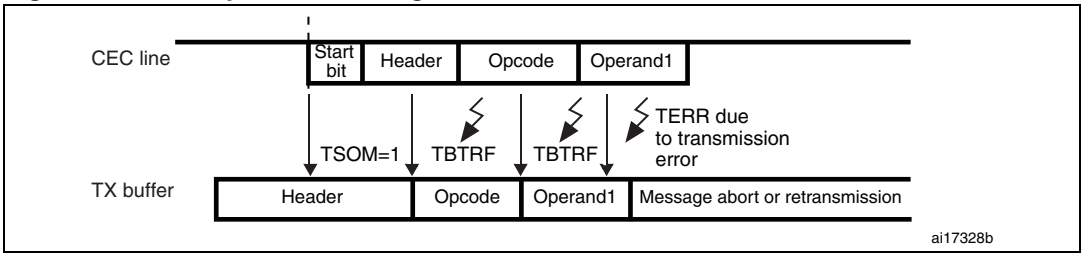
- a directly addressed message block is not acknowledged or a broadcast message block is negatively acknowledged (acknowledge error)
- the TBTRF bit is not cleared while the requested byte needs to be transmitted (TBTFE error)
- an unexpected bit is detected by the bit timing checker/generator (line error)

No error signalling mechanism is specified for the initiator, therefore no specific action is undertaken apart from aborting the current message and clearing the transmit request flag TSOM. The error handler decides if retransmission is possible depending on whether transmission has already failed six times or not. It also sets the transmit request flag if required.

The controller remains in the TX_ERROR state until the transmit error flag TERR is cleared. It then waits for an interframe spacing of 2 bit times before of being ready to process the next message.

An example of a message transmission with errors follows.

Figure 281. Example of a message transmission with transmission error



The software must respect the following sequence in case of a transmission error:

Table 138. Software sequence to respect when transmitting a message

Software sequences	CEC_CSR R/W access	Status bits			
		TBTRF bit 3	TERR bit 2	TEOM bit 1	TSOM bit 0
Write header to TX buffer	-	0	0	0	0
Initiate message transmission by writing TSOM=1	Write 0x01	0	0	0	1
Poll TBTRF or wait until an interrupt occurs	Read 0x00/0x01	0	0	0	1 then 0
The TX buffer is empty (TBTRF is set)	Read 0x08	1	0	0	0
Write opcode to TX buffer	-	1	0	0	0
Acknowledge byte request by writing 0x00	Write 0x00	0	0	0	0
Poll TBTRF or wait until an interrupt occurs	Read 0x00	0	0	0	0
The TX buffer is empty (TBTRF is set)	Read 0x08	1	0	0	0
Write Operand1 to TX buffer	-	1	0	0	0
Acknowledge byte request by writing 0x00	Write 0x00	0	0	0	0
Poll TBTRF or wait until an interrupt occurs	Read 0x00	0	0	0	0
The TX buffer is empty (TBTRF is set)	Read 0x08	1	0	0	0
Write Operand2 to TX buffer	-	1	0	0	0
Acknowledge byte request and signal end of message	Write 0x02	0	0	1	0
Poll TBTRF or wait until an interrupt occurs	Read 0x00	0	0	0	0
Message transmission is completed (TBTRF and TEOM are set)	Read 0x0E	1	1	1	0
If TERR=1, read error code from error status register	-	1	1	1	0
Load header for retransmission if less than 5 retries	-	1	1	1	0
Acknowledge error and possibly request retransmission	Write 0x00 or 0x01	0	0	0	0/1

24.7.8 CEC and system Stop mode

Each time the application does not have anything left to transmit on the CEC line, the system can switch to the low-power Stop mode while waiting for a CEC message.

After wakeup from Stop mode the system runs on the HSI oscillator and the Start bit tolerance is 200 µs. So, the application only has 200 µs to restore the system clock, if needed. You can then choose to clock the system by PLL (with the HSI as the PLL source), or, let the system run on the HSI. In both cases, the CEC prescaler has to be re-adjusted, if needed, before entering the Stop mode.

To switch the system to the low-power Stop mode, and then wake it up on receiving new CEC data without loosing the received data, execute the following procedure:

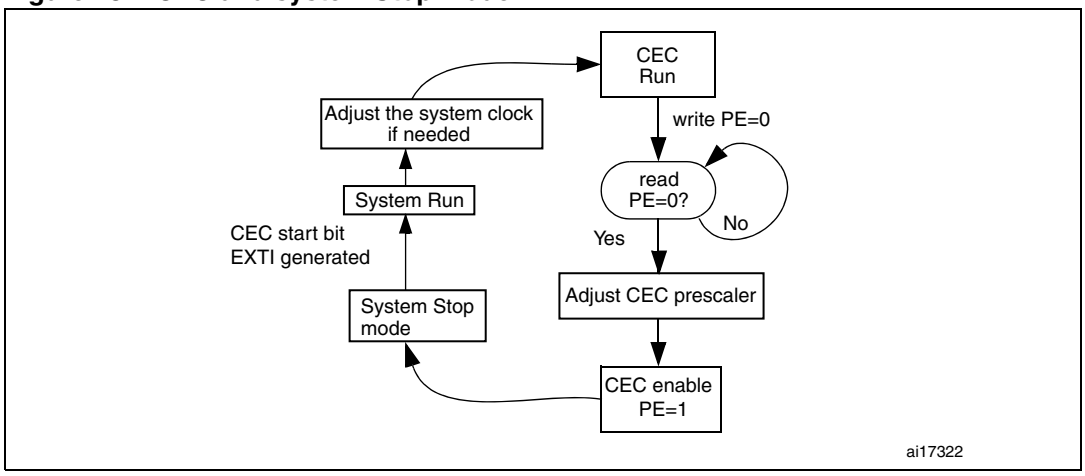
1. Perform a write clear to the peripheral enable (PE) bit in the CEC configuration register, then read back the PE bit value until it is seen at 0. The hardware clears the PE bit

when the current frame (if any) transmission is complete. Once PE is cleared, the HDMI-CEC peripheral is disabled and the CEC line signal is ignored. This operation ensures that the device safely enters the system Stop mode. Otherwise the system clock might stop while the CEC device is in the low impedance state, for example during a handshake bit. In such a case the CEC line gets stuck, causing all of the CEC nodes to be blocked in a stalled condition.

2. Write the CEC prescaler register according to the selected clock you want to use when the system woken up from the Stop mode.
3. Enable the CEC peripheral (PE = 1).
4. Configure the CEC line to generate an external asynchronous interrupt on the falling edge, to wake up the system upon detection of the leading edge of the next start bit.
5. Enter the system Stop mode. The system clock shuts down.
6. When the EXTI falling edge is detected, the system resumes from Stop mode. The STM32F100xx system wakeup is fast enough to allow correct start bit detection.

For more details on CEC wakeup from Stop mode refer to the AN3127: “CEC networking using STM32F100xx value line microcontrollers”.

Figure 282. CEC and system Stop mode



24.8 HDMI-CEC interrupts

An interrupt can be produced:

- during reception if a receive block transfer completes or if a receive error occurs
- during transmission if a Transmit block transfer completes or if a transmit error occurs

Table 139. HDMI-CEC interrupts

Interrupt event	Event flag	Enable control bit
Receive byte/block transfer finished	RBTF	IE
Receive error	RERR	IE
Transmit byte request or block transfer finished	TBTRF	IE
Transmit error	TERR	IE

24.9 HDMI-CEC registers

Refer to *Section 1.1 on page 32* for a list of abbreviations used in register descriptions.

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

24.9.1 CEC configuration register (CEC_CFGR)

This register is used to configure the HDMI-CEC controller.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BPEM	BTEM	IE	PE
												rw	rw	rw	rs

Bits 31:4 Reserved, must be kept cleared.

Bit 3 **BPEM:** Bit period error mode
This bit is set/cleared by software.
0: Standard mode
1: Flexible bit-period mode

Bit 2 **BTEM:** Bit timing error mode
This bit is set/cleared by software.
0: Standard mode
1: Bit timing error-free mode

Bit 1 **IE:** Interrupt enable
This bit is set/cleared by software. It is used to activate an interrupt associated with the set of RTBF, RERR, TBTRF or TERR flags.
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **PE:** Peripheral enable
This bit is set by software, cleared by hardware as soon as the CEC state is Idle.
0: Peripheral disabled
1: Peripheral enabled

24.9.2 CEC own address register (CEC_OAR)

This register is written by the software to define the address of the CEC device.

Address offset: 0x4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												OA[3:0]			
												rw			

Bits 31:4 Reserved, must be kept cleared.

Bits 3:0 **OA[3:0]**: Own address

These bits are written by software to define the own address of the CEC device.

Default value 0x0 is the *TV address* as defined in the HDMI specification.

24.9.3 CEC prescaler register (CEC_PRES)

This register is written by the software to reach the required bit time versus the APB clock frequency.

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PRESC[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept cleared.

Bits 13:0 **PRESC[13:0]**: Prescaler counter value

These bits are written by software to adjust the internal prescaler counter to generate the required 50 µs time base.

PRESC must be selected to respect the following formula: PRESC = 50 x f_{PCLK} (MHz) - 1 where PRESC must be an integer.

Example: if f_{PCLK} = 24 MHz, then PRESC must be programmed to 1199(0x4AF)

24.9.4 CEC error status register (CEC_ESR)

CEC_ESR is the CEC error status register. It contains all the error flags related to the communication.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TBTFE	LINE	ACKE	SBE	RBTFE	BPE	BTE
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept cleared.

Bit 6 **TBTFE***:* Tx block transfer finished error

This bit is set by hardware when the TBTRF bit is not cleared while the requested byte needs to be transmitted.

It is cleared by software when clearing the TX error flag (TERR) of the CEC_CSR register.

Bit 5 **LINE**: Line error

This bit is set by hardware when the CEC line is detected low although it is driven to high impedance while not in the arbitration phase or during the ACK bit.

It is cleared by software by clearing the TX error flag (TERR) in the CEC_CSR register.

Bit 4 **ACKE***:* Block acknowledge error

This bit is set by hardware when a directly addressed message block is not acknowledged or when a broadcast message block is negatively acknowledged.

It is cleared by software when clearing the TX error flag (TERR) or the Rx error flag (RERR) in the CEC_CSR register.

Bit 3 **SBE***:* Start bit error

This bit is set by hardware when the start bit (identified by its low duration only, that is, an error bit), is detected before the end of a message.

It is cleared by software by clearing the Rx error flag (RERR) in the CEC_CSR register.

Bit 2 **RTBFE**: Rx block transfer finished error

This bit is set by hardware when the RBTF bit is not cleared while a new byte is ready to be written to the RX buffer.

It is cleared by software by clearing the Rx error flag (RERR) in the CEC_CSR register.

Bit 1 **BPE**: Bit period error

This bit is set by hardware when the time between two falling edges on the CEC line is too short in Bit period error mode or out of specification in Safe mode, start bit excepted. It is not set if BTE was previously set.

It is cleared by software by clearing the Rx error flag (RERR) in the CEC_CSR register.

Bit 0 **BTE**: Bit timing error

This bit is set by hardware when an incorrect rising edge position is detected on the CEC line while in Safe mode, start bit excepted.

It is cleared by software by clearing the Rx error flag (RERR) in the CEC_CSR register.

24.9.5 CEC control and status register (CEC_CSR)

CEC_CSR is the CEC control & status register. It contains all the flags related to the communication and some control bits to be managed during the communication.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RBTF	RERR	REOM	RSOM	TBTRF	TERR	TEOM	TSOM
								rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	rs

Bits 31:8 Reserved, must be kept cleared.

Bit 7 **RBTF**: Rx byte/block transfer finished

This bit is set by hardware when a data byte has been received. If the IE bit in the CEC_CFGR register is set, an interrupt is generated.

It is cleared by software by writing it to 0 to acknowledge the event and clear the associated interrupt.

Bit 6 **RERR**: Rx error

This bit is set by hardware when a reception error occurs. The software can read the CEC_ESR register to better know which error occurred. If the IE bit in the CEC_CFGR register is set, an interrupt is generated.

It is cleared by software by writing it to 0 to acknowledge the error and clear the associated interrupt.

Bit 5 **REOM**: Rx end of message

This bit is set by hardware when the last date byte of a message has been received.

It is cleared by software by writing it to 0 to acknowledge the event.

Note: Do not clear the REOM bit before the RBTF flag/interrupt is set.

Bit 4 **RSOM**: Rx start of message

This bit is set by hardware when a header byte is received.

It is cleared by software by writing it to 0 to acknowledge the event.

Bit 3 **TBTRF**: Tx byte transfer request or block transfer finished

This bit is set by hardware either to request a new data byte when TEOM is not set or to signal the successful transmission of the message when TEOM is set. If the IE bit in the CEC_CFGR register is set, an interrupt is generated.

It is cleared by software by writing it to 0 to acknowledge the event and clear the associated interrupt.

Bit 2 **TERR**: Tx error

This bit is set by hardware when a transmission error occurs. The software can read the CEC_ESR register to better know which error occurred. If the IE bit in the CEC_CFGR register is set, an interrupt is generated.

It is cleared by software by writing it to 0 to acknowledge the error and clear the associated interrupt.

Bit 1 **TEOM**: Tx end of message

This bit is set and cleared by software except if RTBF or RERR is set.

0: the data byte will be transmitted without any EOM bit
1: the data byte will be transmitted with an EOM bit, signalling the end of the message

Bit 0 **TSOM**: Tx start of message

This bit is set by software to request transmission of a new message. The TX data byte is supposed to contain the header byte.

It is cleared by hardware when access to bus is granted or when a transmission error has occurred.

24.9.6 CEC Tx data register (CEC_TXD)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXD[7:0]							
								rw							

Bits 31:8 Reserved, must be kept cleared.

Bits 7:0 **TXD[7:0]**: Tx Data register.
Data byte to be transmitted.

24.9.7 CEC Rx data register (CEC_RXD)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RXD[7:0]							
								r							

Bits 31:8 Reserved, must be kept cleared.

Bits 7:0 **RXD[7:0]**: Rx data register.
This is a read-only register which contains the last data byte that was received.

24.9.8 HDMI-CEC register map

The following table summarizes the HDMI-CEC registers.

Table 140. HDMI-CEC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CEC_CFGR**	Reserved																												BPEM	BTEM	IE	PE
	Reset value																													0	0	0	0
0x04	**CEC_OAR**	Reserved																												OAR[3:0]			
	Reset value																													0	0	0	0
0x08	**CEC_PRE**	Reserved																		PRESC[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**CEC_ESR**	Reserved																									TBTFE	LINE	ACKE	SBE	RBTFE	BPE	BTE
	Reset value																										0	0	0	0	0	0	0
0x10	**CEC_CSR**	Reserved																								RBTF	RERR	REOM	RSOM	TBTRF	TERR	TEOM	TSOM
	Reset value																									0	0	0	0	0	0	0	0
0x14	**CEC_TXD**	Reserved																								TXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x18	**CEC_RXD**	Reserved																								RXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Table 1: Low and medium-density device register boundary addresses* and *Table 2: High-density device register boundary addresses* for the register boundary addresses.

25 Debug support (DBG)

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to the whole STM32F100xx family, unless otherwise specified.

25.1 Overview

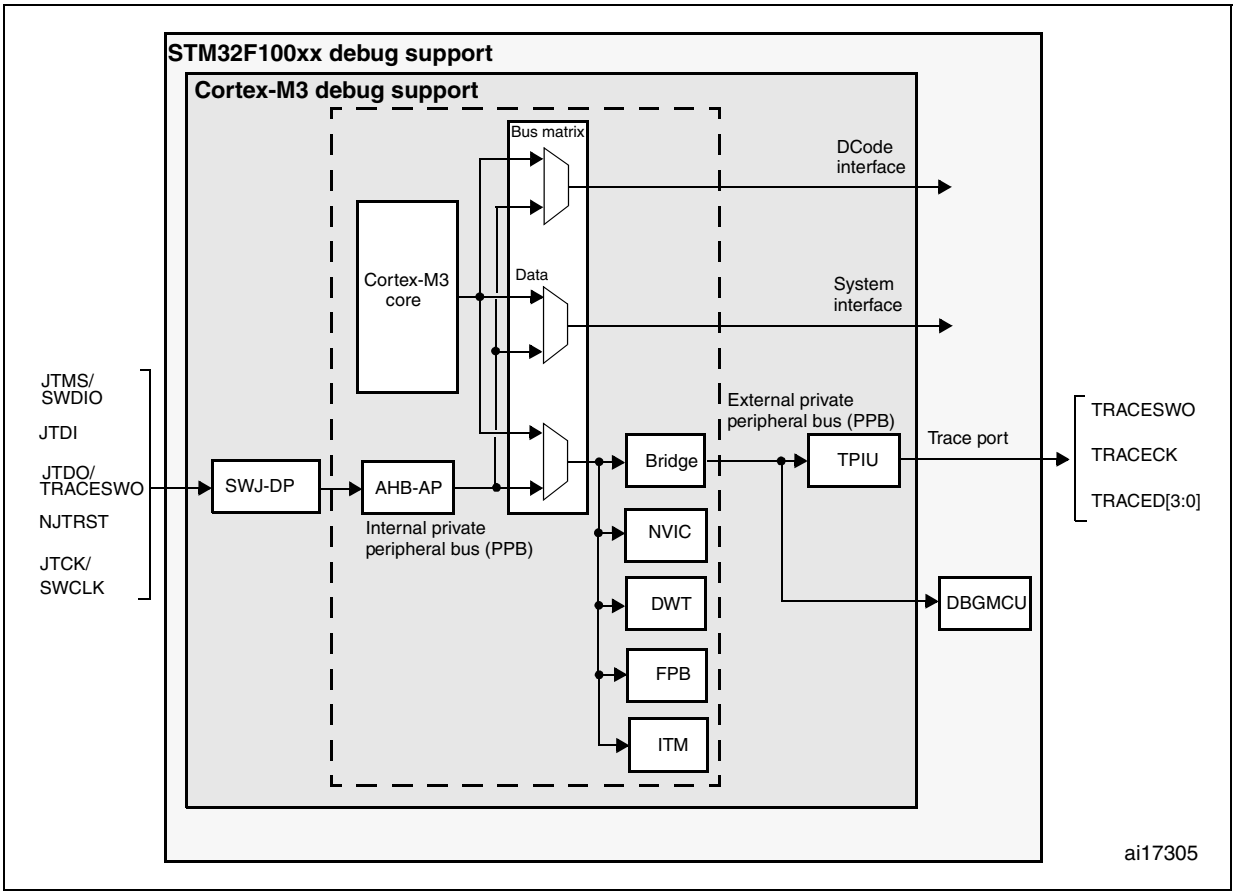
The STM32F100xx are built around a Cortex™-M3 core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32F100xx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 283. Block diagram of STM32 MCU and Cortex™-M3-level debug support



Note: The debug features embedded in the Cortex™-M3 *core are a subset of the ARM CoreSight Design Kit.*

The ARM Cortex™-M3 core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32F100xx:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

*Note: For further information on debug functionality supported by the ARM Cortex™-M3 core, refer to the Cortex™-M3-*r1p1 *Technical Reference Manual and to the CoreSight Design Kit-r1p0 TRM (see Section 25.2: Reference ARM documentation).*

25.2 Reference ARM documentation

- Cortex™-M3 r1p1 Technical Reference Manual (TRM)
  It is available from:
  http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337e/DDI0337E_cortex_m3_r1p1_trm.pdf
- ARM Debug Interface V5
- ARM CoreSight Design Kit revision r1p1 Technical Reference Manual

25.3 SWJ debug port (serial wire and JTAG)

The STM32F100xx core integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an ARM standard CoreSight debug port that combines a JTAG-DP (5-pin) interface and a SW-DP (2-pin) interface.

- The JTAG Debug Port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
- The Serial Wire Debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.

In the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 284. SWJ debug port

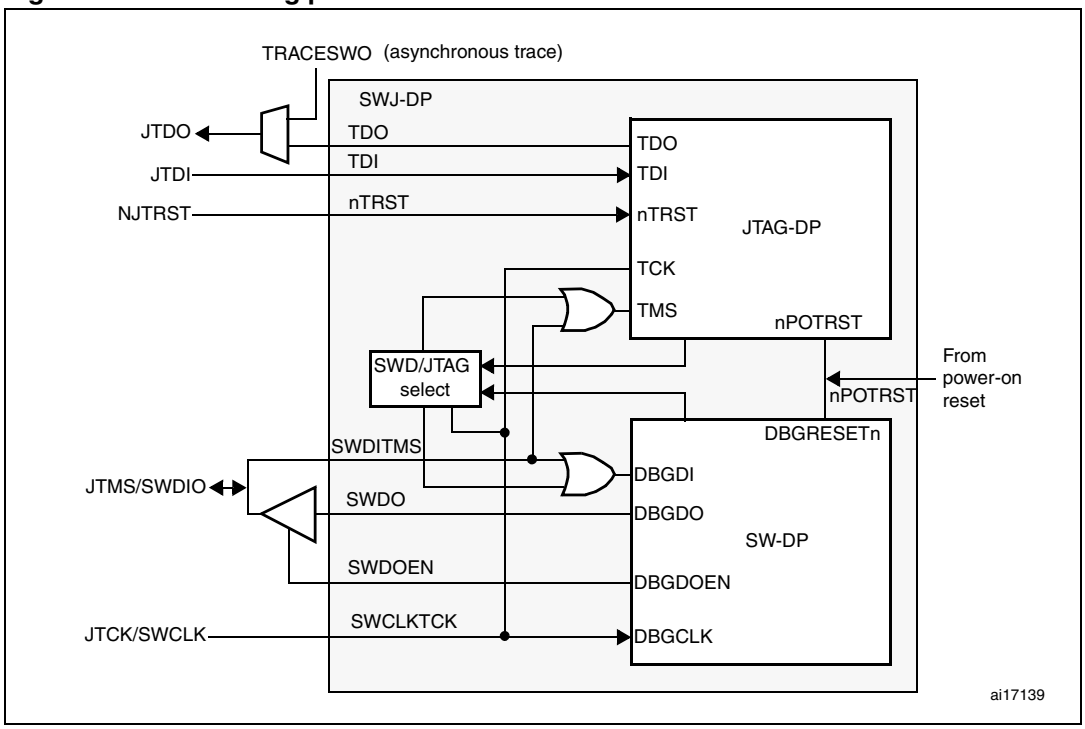


Figure 284 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

25.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

25.4 Pinout and debug port pins

The STM32F100xx MCUs are available in various packages with different numbers of available pins. As a result, some functionality (ETM) related to pin availability may differ between packages.

25.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F100xx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 141. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assign ment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if async trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

25.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, the STM32F100xx MCU implements the *AF remap and debug I/O configuration register (AFIO_MAPR)* register to disable some part or all of the SWJ-DP port and so releases the associated pins for General Purpose IOs usage. This register is mapped on an APB bridge connected to the Cortex™-M3 System Bus. Programming of this register is done by the user software program and not the debugger host.

Three control bits allow the configuration of the SWJ-DP pin assignments. These bits are reset by the System Reset.

- AFIO_MAPR (@ 0x40010004 in the STM32F100xx MCU)
  - READ: APB - No Wait State
  - WRITE: APB - 1 Wait State if the write buffer of the AHB-APB bridge is full.

  Bit 26:24= **SWJ_CFG[2:0]**

  Set and cleared by software.

  These bits are used to configure the number of pins assigned to the SWJ debug port. The goal is to release as much as possible the number of pins to be used as General Purpose IOs if using a small size for the debug port.

  The default state after reset is “000” (whole pins assigned for a full JTAG-DP connection). Only one of the 3 bits can be set (it is forbidden to set more than one bit).

Table 142. Flexible SWJ-DP pin assignment

Available debug ports	SWJ IO pin assigned				
	PA13 / JTMS/ SWDIO	PA14 / JTCK/ SWCLK	PA15 / JTDI	PB3 / JTDO	PB4/ NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset State	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP Disabled and SW-DP Enabled	X	X			
JTAG-DP Disabled and SW-DP Disabled	Released				

Note: When the APB bridge write buffer is full, it takes one extra APB cycle when writing the AFIO_MAPR register. This is because the deactivation of the JTAGSW pins is done in two cycles to guarantee a clean level on the nTRST and TCK input signals of the core.

- Cycle 1: the JTAGSW input signals to the core are tied to 1 or 0 (to 1 for nTRST, TDI and TMS, to 0 for TCK)
- Cycle 2: the GPIO controller takes the control signals of the SWJTAG IO pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

25.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: Internal pull-up
- JTDI: Internal pull-up
- JTMS/SWDIO: Internal pull-up
- TCK/SWCLK: Internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: Input pull-up
- JTDI: Input pull-up
- JTMS/SWDIO: Input pull-up
- JTCK/SWCLK: Input pull-down
- JTDO: Input floating

The software can then use these I/Os as standard GPIOs.

Note: The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.

Having embedded pull-ups and pull-downs removes the need to add external resistors.

25.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must set SWJ_CFG=010 just after reset. This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

Note: For user software designs, note that:

To release the debug pins, remember that they will be first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.

When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.

25.5 STM32F100xx JTAG TAP connection

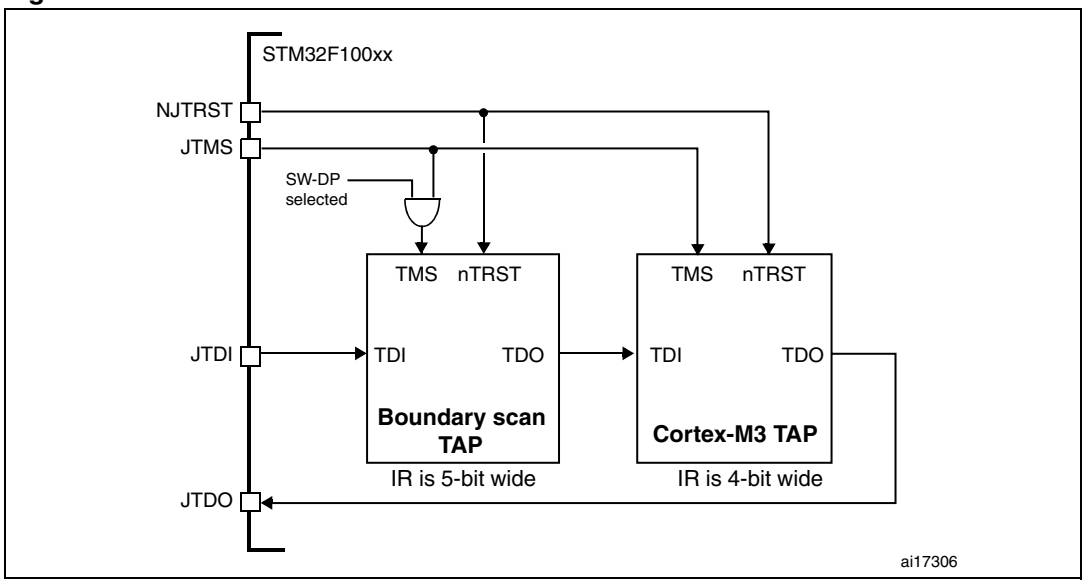
The STM32F100xx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex™-M3 TAP (IR is 4-bit wide).

To access the TAP of the Cortex™-M3 for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

*Note: **Important**: Once Serial-Wire is selected using the dedicated ARM JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).*

Figure 285. JTAG TAP connections



25.6 ID codes and locking mechanism

There are several ID codes inside the STM32F100xx MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

25.6.1 MCU device ID code

The STM32F100xx MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG_MCU component and is mapped on the external PPB bus (see *Section 25.15 on page 658*). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

DBGMCU_IDCODE

Address: 0xE0042000

Only 32-bits access supported. Read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID															
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
Reserved				DEV_ID											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID(15:0)** Revision identifier

This field indicates the revision of the device:

– 0x1000 = Revision A

– 0x1001 = Revision Z

Bits 31:16 **REV_ID(15:0)** Revision identifier
This field indicates the revision of the device:
In low and medium density value line devices:
– 0x1000 = Revision A
– 0x1001 = Revision Z
– In high density value line devices:
– 0x1000 = Revision A

Bits 15:12 Reserved

Bits 11:0 **DEV_ID(11:0)**: Device identifier
This field indicates the device ID.
The device ID is 0x420

Bits 11:0 **DEV_ID(11:0)**: Device identifier
This field indicates the device ID.
For low- and medium-density value line devices, the device ID is 0x420
For high-density value line devices, the device ID is 0x428

25.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32F100xx BSC (boundary scan) integrates a JTAG ID code equal to

- In low and medium-density value line devices:
  – 0x06420041 = Revision A and Revision Z
- In high-density value line devices:
  – 0x06428041 = Revision A

25.6.3 Cortex™-M3 TAP

The TAP of the ARM Cortex™-M3 integrates a JTAG ID code. This ID code is the ARM default one and has not been modified. This code is only accessible by the JTAG Debug Port.
This code is **0x3BA00477** (corresponds to Cortex™-M3 r1p1-01rel0, see *Section 25.2: Reference ARM documentation*).

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

25.6.4 Cortex™-M3 JEDEC-106 ID code

The ARM Cortex™-M3 integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00FF000_0xE00FFFFF.

This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

25.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex™-M3r1p1 Technical Reference Manual (TRM), for references, please see *Section 25.2: Reference ARM documentation*).

Table 143. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	*ID CODE* 0x3BA00477 (ARM Cortex™-M3 r1p1-01rel0 ID Code)
1010	DPACC [35 bits]	*Debug port access register* This initiates a debug port and allows 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 Bits 2:1 = A[3:2] = 2-bit address of a debug 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 which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved Refer to *Table 144* for a description of the A(3:2) bits
1011	APACC [35 bits]	*Access port access register* Initiates an access port and allows 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 address (sub-address AP registers). Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved There are many AP Registers (see AHB-AP) addressed as the combination of: – The shifted value A[3:2] – The current value of the DP SELECT register
1000	ABORT [35 bits]	*Abort register* – Bits 31:1 = Reserved – Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 144. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x0	00	Reserved
0x4	01	DP CTRL/STAT register. Used to: – Request a system or debug power-up – Configure the transfer operation for AP accesses – Control the pushed compare and pushed verify operations. – Read some status flags (overrun, power-up acknowledges)
0x8	10	DP SELECT register: Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

25.8 SW debug port

25.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 KΩ recommended by ARM).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

25.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

Table 145. Packet request (8-bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request
4:3	A(3:2)	Address field of the DP or AP registers (refer to *Table 144*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as “1” by the target because of the pull-up

Refer to the Cortex™-M3 r1p1 *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

Table 146. ACK response (3 bits)

Bit	Name	Description
0..2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

Table 147. DATA transfer (33 bits)

Bit	Name	Description
0..31	WDATA or RDATA	Write or Read data
32	Parity	Single parity of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

25.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default ARM one and is set to **0x1BA01477** (corresponding to Cortex™-M3 r1p1).

Note: Note that the SW-DP state machine is inactive until the target reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target will issue a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex™-M3 r1p1 TRM* and the *CoreSight Design Kit r1p0 TRM*.

25.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.
  The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)
  This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it will fail.

25.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 148. SW-DP registers

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read		IDCODE	The manufacturer code is not set to ST code. **0x1BA01477** (identifies the SW-DP)
00	Write		ABORT	

Table 148. SW-DP registers (continued)

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read		READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write		SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write		READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

25.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

25.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHP-AP resisters are 6-bits wide (up to 64 words or 256 bytes) and consists of:

d) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register

e) Bits [3:2] = the 2 address bits of A(3:2) of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex™-M3 includes 9 x 32-bits registers:

Table 149. Cortex™-M3 AHB-AP registers

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type
0x04	AHB-AP Transfer Address	
0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address Register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID Register	

Refer to the *Cortex™-M3 r1p1 TRM* for further details.

25.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

Table 150. Core debug registers

Register	Description
DHCSR	*The 32-bit Debug Halting Control and Status Register* This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	*The 17-bit Debug Core Register Selector Register:* This selects the processor register to transfer data to or from.
DCRDR	*The 32-bit Debug Core Register Data Register:* This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	*The 32-bit Debug Exception and Monitor Control Register:* This provides Vector Catching and Debug Monitor Control. This register contains a bit named ***TRCENA*** which enable the use of a TRACE.

Note: ***Important****: these registers are not reset by a system reset. They are only reset by a power-on reset.*

Refer to the *Cortex™-M3* r1p*1 TRM* for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C_DEBUGEN) of the Debug Halting Control and Status Register.

25.11 Capability of the debugger host to connect under system reset

The STM32F100xx MCUs' reset system comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up.
- Internal watchdog reset
- Software reset
- External reset

The Cortex™-M3 differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn)

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug Registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core will immediately halt without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

Note: *It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.*

25.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space.
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.

25.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead

25.14 ITM (instrumentation trace macrocell)

25.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex™-M3 clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before you program or use the ITM.

25.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80_00_00_00_00_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control Register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control Register must be set.

Note: If the SYNENA bit is not set, the DWT generates Synchronization triggers to the TPIU which will send only TPIU synchronization packets and not ITM synchronization packets.

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

Table 151. Main ITM registers

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data.
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock).
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets.
		Bit 1 = TSENA (Timestamp Enable)
		Bit 0 = ITMENA: Global Enable Bit of the ITM
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace.
@E0000000-E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out.

Example of configuration

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU_CR (refer to *Section 25.16.2: TRACE pin assignment* and *Section 25.15.3: Debug MCU configuration register*)
- Write 0xC5ACCE55 to the ITM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control Register to enable the ITM with Sync enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable Register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege Register to unmask stimulus ports 7:0
- Write the value to output in the Stimulus Port Register 0: this can be done by software (using a printf function)

25.15 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I2C during a breakpoint
- Control of the trace pins assignment

25.15.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

25.15.2 Debug support for timers, watchdog and I^2C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I^2C, the user can choose to block the SMBUS timeout during a breakpoint.

25.15.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support
- Timer and watchdog counter support
- Trace pin assignment

This DBGMCU_CR is mapped on the External PPB bus at address 0xE0042004

It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

DBGMCU_CR

Address: 0xE004 2004

Only 32-bit access supported

POR Reset: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	Res.	DBG_I2C2_SMBUS_TIMEOUT
				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
DBG_I2C1_SMBUS_TIMEOUT	Res.	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG STOP	TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

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

Bits 27:25 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x = 12..14)

0: The clock of the involved timer counter is fed even if the core is halted

1: The clock of the involved timer counter is stopped when the core is halted

Bits 24:22 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x = 17..15)

0: The clock of the involved timer counter is fed even if the core is halted

1: The clock of the involved timer counter is stopped when the core is halted

Bit 21 Reserved, must be kept cleared.

Bits 20:18 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=5 .. 7)

0: The clock of the involved timer counter is fed even if the core is halted, and the outputs behave normally.

1: The clock of the involved timer counter is stopped when the core is halted, and the outputs are disabled (as if there were an emergency stop in response to a break event).

Bit 17 Reserved, must be kept cleared.

Bit 16 **DBG_I2C2_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when Core is halted

0: Same behavior as in normal mode

1: The SMBUS timeout is frozen

Bit 15 **DBG_I2C1_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when Core is halted
0: Same behavior as in normal mode
1: The SMBUS timeout is frozen

Bits 13:10 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=4..1)

Bit 9 **DBG_WWDG_STOP:** Debug window watchdog stopped when core is halted
0: The window watchdog counter clock continues even if the core is halted
1: The window watchdog counter clock is stopped when the core is halted

Bit 8 **DBG_IWDG_STOP:** Debug independent watchdog stopped when core is halted
0: The watchdog counter clock continues even if the core is halted
1: The watchdog counter clock is stopped when the core is halted

Bits 7:5 **TRACE_MODE[1:0] and TRACE_IOEN**: Trace pin assignment control
– *With TRACE_IOEN=0:*
TRACE_MODE=xx: TRACE pins not assigned (default state)
– *With TRACE_IOEN=1:*
  – TRACE_MODE=00: TRACE pin assignment for Asynchronous Mode
  – TRACE_MODE=01: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
  – TRACE_MODE=10: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
  – TRACE_MODE=11: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY:** Debug Standby mode
0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.
From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)
1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset

Bit 1 **DBG_STOP:** Debug Stop mode
0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.
1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG_STOP=0)

Bit 0 **DBG_SLEEP:** Debug Sleep mode
0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.
In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.
1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

25.16 TPIU (trace port interface unit)

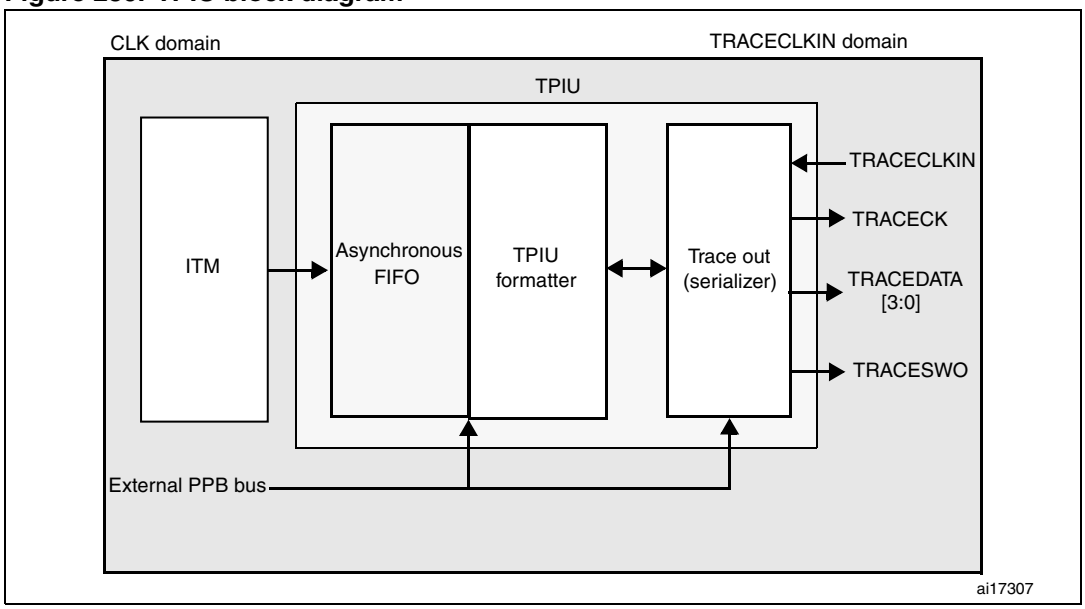
25.16.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Figure 286. TPIU block diagram



25.16.2 TRACE pin assignment

- Asynchronous mode

  The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

Table 152. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F100xx pin assignment
	Type	Description	
TRACESWO	O	TRACE Async Data Output	PB3

- Synchronous mode

  The synchronous mode requires from 2 to 6 extra pins depending on the data trace size and is only available in the larger packages. In addition it is available in JTAG mode and in Serial Wire mode and provides better bandwidth output capabilities than asynchronous trace.

Table 153. Synchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F100xx pin assignment
	Type	Description	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

TPUI TRACE pin assignment

By default, these pins are NOT assigned. They can be assigned by setting the TRACE_IOEN and TRACE_MODE bits in the ***MCU Debug component configuration register***. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode**: 1 extra pin is needed
- **Synchronous mode**: from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4):
  - TRACECK
  - TRACED(0) if port size is configured to 1, 2 or 4
  - TRACED(1) if port size is configured to 2 or 4
  - TRACED(2) if port size is configured to 4
  - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE_IOEN and TRACE_MODE[1:0] of the Debug MCU configuration Register (DBGMCU_CR). By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

Table 154. Flexible TRACE pin assignment

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned					
TRACE_IOEN	TRACE_MODE[1:0]		**PB3 /** JTDO/ TRACESWO	**PE2 /** TRACECK	**PE3 /** TRACED[0]	**PE4 /** TRACED[1]	**PE5 /** TRACED[2]	**PE6 /** TRACED[3]
0	XX	No Trace (default state)	Released (1)					
1	00	Asynchronous Trace	TRACESWO			Released (usable as GPIO)		
1	01	Synchronous Trace 1 bit	Released (1)	TRACECK	TRACED[0]			
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]		
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. When Serial Wire mode is used, it is released. But when JTAG is used, it is assigned to JTDO.

Note: By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE_IOEN has been set.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS_R (Current Sync Port Size Register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

25.16.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
  - 1 bit (LSB) to indicate it is a DATA byte (‘0) or an ID byte (‘1).
  - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
  - if the corresponding byte was a data, this bit gives bit0 of the data.
  - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

Note: Refer to the ARM CoreSight Architecture Specification v1.0 (ARM IHI 0029B) for further information

25.16.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)

  It consists of the word: 0x7F_FF_FF_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.

  It is output periodically ***between*** frames.

  In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.
- The Half-Word Synchronization packet

  It consists of the half word: 0x7F_FF (LSB emitted first).

  It is output periodically ***between or within*** frames.

  These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

25.16.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core. Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The TPUI Frame synchronization packet (0x7F_FF_FF_FF) is emitted:

- after each TPIU reset release. This reset is synchronously released with the rising edge of the TRACECLKIN clock. This means that this packet is transmitted when the TRACE_IOEN bit in the DBGMCU_CFG register is set. In this case, the word 0x7F_FF_FF_FF is not followed by any formatted packet.
- at each DWT trigger (assuming DWT has been previously configured). Two cases occur:
  - If the bit SYNENA of the ITM is reset, only the word 0x7F_FF_FF_FF is emitted without any formatted stream which follows.
  - If the bit SYNENA of the ITM is set, then the ITM synchronization packets will follow (0x80_00_00_00_00_00), formatted by the TPUI (trace source ID added).

25.16.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

Note: In this synchronous mode, it is not required to provide a stable clock frequency.

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

25.16.7 Asynchronous mode

This is a low cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32F100xx packages.

This asynchronous mode requires a constant frequency for TRACECLKIN. For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.

25.16.8 TRACECLKIN connection inside the STM32F100xx

In the STM32F100xx, this TRACECLKIN input is internally connected to HCLK. This means that when in asynchronous trace mode, the application is restricted to use to time frames where the CPU frequency is stable.

Note: ***Important:*** *when using asynchronous trace: it is important to be aware that:*

The default clock of the STM32F100xx MCUs is the internal RC oscillator. Its frequency under reset is different from the one after reset release. This is because the RC calibration is the default one under system reset and is updated at each system reset release.

Consequently, the trace port analyzer (TPA) should not enable the trace (with the TRACE_IOEN bit) under system reset, because a Synchronization Frame Packet will be issued with a different bit time than trace packets which will be transmitted after reset release.

25.16.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control Register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

Table 155. Important TPIU registers

Address	Register	Description
0xE0040004	Current port size	Allows the trace port size to be selected: Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4 Only 1 bit must be set. By default, the port size is one bit. (0x00000001)
0xE00400F0	Selected pin protocol	Allows the Trace Port Protocol to be selected: Bit1:0= 00: Sync Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved
0xE0040304	Formatter and flush control	Bit 31-9 = always ‘0 Bit 8 = TrigIn = always ‘1 to indicate that triggers are indicated Bit 7-4 = always 0 Bit 3-2 = always 0 Bit 1 = EnFCont. In Sync Trace mode (Select_Pin_Protocol register bit1:0=00), this bit is forced to ‘1: the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 <> 00), this bit can be written to activate or not the formatter. Bit 0 = always 0 The resulting default value is 0x102 **Note:** In synchronous mode, because the TRACECTL pin is not mapped outside the chip, the formatter is always enabled in continuous mode -this way the formatter inserts some control packets to identify the source of the trace packets).
0xE0040300	Formatter and flush status	Not used in Cortex™-M3, always read as 0x00000008

25.16.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control Register (DEMCR)
- Write the TPIU Current Port Size Register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control Register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the sync or async mode. Example: 0x2 for async NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO_TRACEN) to assign TRACE I/Os for async mode. A TPIU Sync packet is emitted at this time (FF_FF_FF_7F)
- Configure the ITM and write the ITM Stimulus register to output a value

25.17 DBG register map

The following table summarizes the Debug registers

Table 156. Value DBG register map and reset values

Addr.	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
0xE0042000	DBGMCU_IDCODE	REV_ID																Reserved				DEV_ID											
	Reset value[1]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0xE0042004	DBGMCU_CR	Reserved				DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Reserved	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	Reserved	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDGSTOP	TRACE_MODE[1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	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

1. The reset value is product dependent. For more information, refer to *Section 25.6.1: MCU device ID code*.

26 Device electronic signature

Low-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density value line devices are STM32F100xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

This section applies to all STM32F100xx devices, unless otherwise specified.

The electronic signature is stored in the System memory area in the Flash memory module, and can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the STM32F100xx microcontroller.

26.1 Memory size registers

26.1.1 Flash size register

Base address: 0x1FFF F7E0

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **F_SIZE:** Flash memory size

This field value indicates the Flash memory size of the device in Kbytes.
Example: 0x0080 = 128 Kbytes.

26.2 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

Base address: 0x1FFF F7E8

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
U_ID(15:0)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **U_ID(15:0):** 15:0 unique ID bits

Address offset: 0x02

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(31:16)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **U_ID(31:16):** 31:16 unique ID bits
This field value is also reserved for a future feature.

Address offset: 0x04

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
U_ID(63:48)															
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
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(63:32):** 63:32 unique ID bits

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
U_ID(95:80)															
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
U_ID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(95:64):** *95:64 Unique ID bits.*

Index

A

ADC_CR1 172
ADC_CR2 173
ADC_DR 183
ADC_HTR 178
ADC_JDRx 183
ADC_JOFRx 177
ADC_JSQR 182
ADC_LTR 178
ADC_SMPR1 176
ADC_SMPR2 177
ADC_SQR1 179
ADC_SQR2 180
ADC_SQR3 181
ADC_SR 171
AFIO_EVCR 120
AFIO_EXTICR1 123
AFIO_EXTICR2 123
AFIO_EXTICR3 124
AFIO_EXTICR4 124
AFIO_MAPR 121
AFIO_MAPR2 125

B

BKP_CR 65
BKP_CSR 65
BKP_DRx 64
BKP_RTCCR 64

C

CEC_CFGR 634
CEC_CSR 637
CEC_ESR 636
CEC_OAR 635
CEC_PRES 635
CEC_RXD 638
CEC_TXD 638
CRC_DR 47
CRC_IDR 47

D

DAC_CR 197
DAC_DHR12L1 201
DAC_DHR12L2 202
DAC_DHR12LD 203
DAC_DHR12R1 201
DAC_DHR12R2 202
DAC_DHR12RD 203
DAC_DHR8R1 201
DAC_DHR8R2 202
DAC_DHR8RD 204
DAC_DOR1 204
DAC_DOR2 204
DAC_SR 205
DAC_SWTRIGR 200
DBGMCU_CR 659
DBGMCU_IDCODE 647
DMA_CCRx 153
DMA_CMARx 155
DMA_CNDTRx 154
DMA_CPARx 155
DMA_IFCR 152
DMA_ISR 151

E

EXTI_EMR 136
EXTI_FTSR 137
EXTI_IMR 136
EXTI_PR 138
EXTI_RTSR 137
EXTI_SWIER 138

F

FSMC_BCR1..4 506
FSMC_BTR1..4 508
FSMC_BWTR1..4 510

G

GPIOx_BRR 113
GPIOx_BSRR 112
GPIOx_CRH 111
GPIOx_CRL 110
GPIOx_IDR 111
GPIOx_LCKR 113
GPIOx_ODR 112

I

I2C_CCR 573
I2C_CR1 563
I2C_CR2 565

I2C_DR 568
I2C_OAR1 567
I2C_OAR2 567
I2C_SR1 568
I2C_SR2 572
I2C_TRISE 574
IWDG_KR 466
IWDG_PR 467
IWDG_RLR 467
IWDG_SR 468

P

PWR_CR 58
PWR_CSR 60

R

RCC_AHBENR 87
RCC_APB1ENR 91
RCC_APB1RSTR 85
RCC_APB2ENR 89
RCC_APB2RSTR 83
RCC_BDCR 94
RCC_CFGR 79
RCC_CFGR2 97
RCC_CIR 81
RCC_CR 77
RCC_CSR 95
RTC_ALRH 462
RTC_ALRL 462
RTC_CNTH 461
RTC_CNTL 461
RTC_CRH 457
RTC_CRL 458
RTC_DIVH 460
RTC_DIVL 460
RTC_PRLH 459
RTC_PRLL 460

S

SPI_CR1 535
SPI_CR2 536
SPI_CRCPR 539
SPI_DR 538
SPI_RXCRCR 540
SPI_SR 538
SPI_TXCRCR 540

T

TIM15_ARR 415
TIM15_BDTR 417
TIM15_CCER 412
TIM15_CCMR1 409
TIM15_CCR1 416
TIM15_CCR2 417
TIM15_CNT 415
TIM15_CR1 402
TIM15_CR2 403
TIM15_DCR 419
TIM15_DIER 406
TIM15_DMAR 419
TIM15_EGR 408
TIM15_PSC 415
TIM15_RCR 416
TIM15_SMCR 404
TIM15_SR 407
TIMx_ARR 326, 363, 372, 450
TIMx_BDTR 268, 434
TIMx_CCER 262, 324, 362, 371, 430
TIMx_CCMR1 . . . . . .258, 320, 359, 369, 427
TIMx_CCMR2 . . . . . .261, 323
TIMx_CCR1 . . . . . .266, 327, 364, 373, 433
TIMx_CCR2 . . . . . .267, 327, 364
TIMx_CCR3 . . . . . .267, 328
TIMx_CCR4 . . . . . .268, 328
TIMx_CNT 264, 325, 363, 372, 432, 449
TIMx_CR1 . . . . . .248, 311, 353, 366, 422, 446
TIMx_CR2 . . . . . .249, 313, 354, 423, 448
TIMx_DCR 270, 329, 435
TIMx_DIER 253, 316, 356, 367, 424, 448
TIMx_DMAR 271, 329, 436
TIMx_EGR 256, 319, 358, 368, 426, 449
TIMx_PSC 265, 326, 363, 372, 432, 450
TIMx_RCR 266, 433
TIMx_SMCR 251, 314, 355
TIMx_SR 255, 317, 357, 367, 425, 449

U

USART_BRR 610
USART_CR1 611
USART_CR2 613
USART_CR3 614
USART_DR 610
USART_GTPR 616
USART_SR 608

W

WWDG_CFR 474
WWDG_CR 473
WWDG_SR 474

27 Revision history

Table 157. Document revision history

Date	Revision	Changes
26-Feb-2010	1	Initial release.
04-Jun-2010	2	Corrected description of TIMx_CCER register in *Section 12.4.9 on page 262* and *Section 13.4.9 on page 324* Updated *Section 14.4.5: Input capture mode on page 343* Added method 1 and 2 in *Section 22.3.3: I2C master mode* Updated note in POS bit description *Section 22.6: I2C registers*
12-Oct-2010	3	Updated for high density value line devices Updated *Section 20.5.2: Supported memories and transactions on page 483* Added *Section 14: General-purpose timers (TIM12/13/14) on page 333* Added *Section 20: Flexible static memory controller (FSMC) on page 476*
21-Jul-2011	4	Corrected *Figure 2: High density value line system architecture on page 34* Updated SPI table in *Section 7.1.11: GPIO configurations for device peripherals on page 106* Updated bit descriptions in *Section 7.3.1: Clock control register (RCC_CR) on page 96* and *Section 8.3.1: Clock control register (RCC_CR) on page 129* EXTI: Updated *Figure 18: External interrupt/event controller block diagram* **ADC**: Corrected *Table 59: External trigger for regular channels for ADC1* and *Table 60: External trigger for injected channels for ADC1 on page 167* **TIMERS:** Removed wrong references to 32-bit counter in *Section 13.4: TIMx2 to TIM5 registers on page 311* TIM1&TIM8: Updated example and definition of DBL bits in *Section 12.4.19: TIM1 DMA control register (TIMx_DCR)*. Added example related to DMA burst feature and description of DMAB bits in *Section 12.4.20: TIM1 DMA address for full transfer (TIMx_DMAR)*. TIM2 to TIM5 and TIM15 to 17: added example and updated definition of DBL bits in *Section 13.4.17: TIMx DMA control register (TIMx_DCR)*. Added example related to DMA burst feature and description of DMAB bits in *Section 13.4.18: TIMx DMA address for full transfer (TIMx_DMAR)*. Updated definition of DBL bits in *Section 13.4.17: TIMx DMA control register (TIMx_DCR)*. In *Section 12.3.3: Repetition counter* Added paragraph “In Center aligned mode, for odd values of RCR, ...” Modified *Figure 167: Update rate examples depending on mode and TIMx_RCR register settings on page 384*

Table 157. Document revision history (continued)

Date	Revision	Changes
21-Jul-2011	4 continued	**WWDG** Updated *Section 19.2: WWDG main features*. Updated *Section 19.3: WWDG functional description* to remove paragraph related to counter reload using EWI interrupt. **I2C:** Updated BERR bit description in *Section 22.6.6: I2C Status register 1 (I2C_SR1)*. Updated *Note 1* in *Section 22.6.8: I2C Clock control register (I2C_CCR)*. Added note 3 below *Figure 235: Transfer sequence diagram for slave transmitter on page 546*. Added note below *Figure 236: Transfer sequence diagram for slave receiver on page 547*. Modified *Section : Closing slave communication on page 547*. Modified STOPF, ADDR, bit description in *Section 22.6.6: I2C Status register 1 (I2C_SR1) on page 568*. Modified *Section 22.6.7: I2C Status register 2 (I2C_SR2) on page 572*. **USART:** Updated *Figure 224: Mute mode using address mark detection* for Address =1.**SPI:** Modified *Slave select (NSS) pin management on page 516* and note on NSS in *Section 21.3.3: Configuring the SPI in master mode* **FSMC:** Updated description of DATLAT , DATAST , and ADDSET bits in *SRAM/NOR-Flash chip-select timing registers 1..4 (FSMC_BTR1..4) on page 508*. Updated byte select description in *Section 20.5.2: Supported memories and transactions on page 483*

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2011 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com