

## Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32F373Cx/Rx/Vx and STM32F378Cx/Rx/Vx microcontroller memory and peripherals. The STM32F373Cx/Rx/Vx and STM32F378Cx/Rx/Vx will be referred to as STM32F37xx throughout the document, unless otherwise specified.

The STM32F37xx is a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics please refer to the STM32F37xx datasheet.

For information on the ARM Cortex<sup>®</sup>-M4 core with FPU, please refer to the STM32F3xx/STM32F4xx programming manual (PM0214).

## Related documents

- STM32F373xx and STM32F378xx datasheet available from your nearest ST sales office.
- STM32F3xx/F4xx Cortex<sup>®</sup>-M4 programming manual (PM0214) available from <http://www.st.com>.

# Contents

<b>1</b>	<b>Documentation conventions</b>	<b>36</b>
1.1	List of abbreviations for registers	36
1.2	Glossary	36
1.3	Peripheral availability	36
<b>2</b>	<b>System architecture and memory overview</b>	<b>37</b>
2.1	System architecture	37
2.1.1	S0: I-bus	37
2.1.2	S1: D-bus	38
2.1.3	S2: S-bus	38
2.1.4	S3, S4: DMA-bus	38
2.1.5	BusMatrix-S (5M5S)	38
2.2	Memory organization	39
2.2.1	Introduction	39
2.2.2	Memory map and register boundary addresses	39
2.3	Embedded SRAM	43
2.3.1	<b>Parity check</b>	<b>43</b>
2.4	Bit banding	43
2.5	Flash memory overview	44
2.6	Boot configuration	44
<b>3</b>	<b>Embedded Flash memory</b>	<b>46</b>
3.1	Flash main features	46
3.2	Flash memory functional description	46
3.2.1	Flash memory organization	46
3.2.2	Read operations	47
3.2.3	Flash program and erase operations	49
3.3	Memory protection	55
3.3.1	Read protection	55
3.3.2	Write protection	57
3.3.3	Option byte block write protection	58
3.4	Flash interrupts	58
3.5	Flash register description	59

3.5.1	Flash access control register (FLASH_ACR) .....	59
3.5.2	Flash key register (FLASH_KEYR) .....	59
3.5.3	Flash option key register (FLASH_OPTKEYR) .....	60
3.5.4	Flash status register (FLASH_SR) .....	60
3.5.5	Flash control register (FLASH_CR) .....	61
3.5.6	Flash address register (FLASH_AR) .....	62
3.5.7	Option byte register (FLASH_OBR) .....	63
3.5.8	Write protection register (FLASH_WRP) .....	64
3.6	Flash register map .....	64
<b>4</b>	<b>Option byte description .....</b>	<b>66</b>
<b>5</b>	<b>Cyclic redundancy check calculation unit (CRC) .....</b>	<b>69</b>
5.1	Introduction .....	69
5.2	CRC main features .....	69
5.3	CRC functional description .....	70
5.4	CRC registers .....	71
5.4.1	Data register (CRC_DR) .....	71
5.4.2	Independent data register (CRC_IDR) .....	72
5.4.3	Control register (CRC_CR) .....	72
5.4.4	Initial CRC value (CRC_INIT) .....	73
5.4.5	CRC polynomial (CRC_POL) .....	73
5.4.6	CRC register map .....	74
<b>6</b>	<b>Power control (PWR) .....</b>	<b>75</b>
6.1	Power supplies .....	75
6.1.1	Independent A/D and D/A converter supply and reference voltage .....	77
6.1.2	Battery backup domain .....	78
6.1.3	Voltage regulator .....	79
6.2	Power supply supervisor .....	79
6.2.1	Power on reset (POR)/power down reset (PDR) .....	79
6.2.2	Programmable voltage detector (PVD) .....	81
6.2.3	External NPOR signal .....	81
6.3	Low-power modes .....	82
6.3.1	Slowing down system clocks .....	82
6.3.2	Peripheral clock gating .....	83

- 6.3.3 Sleep mode ..... 83
- 6.3.4 Stop mode ..... 84
- 6.3.5 Standby mode ..... 86
- 6.3.6 Auto-wakeup from low-power mode ..... 87
- 6.4 Power control registers ..... 89
  - 6.4.1 Power control register (PWR\_CR) ..... 89
  - 6.4.2 Power control/status register (PWR\_CSR) ..... 91
  - 6.4.3 PWR register map ..... 93
- 7 Reset and clock control (RCC) ..... 94**
  - 7.1 Reset ..... 94
    - 7.1.1 Power reset ..... 94
    - 7.1.2 System reset ..... 94
    - 7.1.3 RTC domain reset ..... 95
  - 7.2 Clocks ..... 96
    - 7.2.1 HSE clock ..... 99
    - 7.2.2 HSI clock ..... 101
    - 7.2.3 PLL ..... 101
    - 7.2.4 LSE clock ..... 101
    - 7.2.5 LSI clock ..... 102
    - 7.2.6 System clock (SYSCLK) selection ..... 102
    - 7.2.7 Clock security system (CSS) ..... 103
    - 7.2.8 ADC clock ..... 103
    - 7.2.9 SDADC clock ..... 103
    - 7.2.10 RTC clock ..... 103
    - 7.2.11 Watchdog clock ..... 104
    - 7.2.12 Clock-out capability ..... 104
    - 7.2.13 Internal/external clock measurement using TIM14 ..... 105
  - 7.3 Low-power modes ..... 106
  - 7.4 RCC registers ..... 107
    - 7.4.1 Clock control register (RCC\_CR) ..... 107
    - 7.4.2 Clock configuration register (RCC\_CFGR) ..... 109
    - 7.4.3 Clock interrupt register (RCC\_CIR) ..... 113
    - 7.4.4 APB2 peripheral reset register (RCC\_APB2RSTR) ..... 115
    - 7.4.5 APB1 peripheral reset register (RCC\_APB1RSTR) ..... 117
    - 7.4.6 AHB peripheral clock enable register (RCC\_AHBENR) ..... 119
    - 7.4.7 APB2 peripheral clock enable register (RCC\_APB2ENR) ..... 121

7.4.8	APB1 peripheral clock enable register (RCC_APB1ENR) . . . . .	123
7.4.9	RTC domain control register (RCC_BDCR) . . . . .	126
7.4.10	Control/status register (RCC_CSR) . . . . .	128
7.4.11	AHB peripheral reset register (RCC_AHBRSR) . . . . .	130
7.4.12	Clock configuration register 2 (RCC_CFGR2) . . . . .	132
7.4.13	Clock configuration register 3 (RCC_CFGR3) . . . . .	133
7.4.14	RCC register map . . . . .	135
<b>8</b>	<b>General-purpose I/Os (GPIO) . . . . .</b>	<b>137</b>
8.1	Introduction . . . . .	137
8.2	GPIO main features . . . . .	137
8.3	GPIO functional description . . . . .	137
8.3.1	General-purpose I/O (GPIO) . . . . .	139
8.3.2	I/O pin alternate function multiplexer and mapping . . . . .	140
8.3.3	I/O port control registers . . . . .	141
8.3.4	I/O port data registers . . . . .	141
8.3.5	I/O data bitwise handling . . . . .	141
8.3.6	GPIO locking mechanism . . . . .	141
8.3.7	I/O alternate function input/output . . . . .	142
8.3.8	External interrupt/wakeup lines . . . . .	142
8.3.9	Input configuration . . . . .	142
8.3.10	Output configuration . . . . .	143
8.3.11	Alternate function configuration . . . . .	144
8.3.12	Analog configuration . . . . .	145
8.3.13	Using the HSE or LSE oscillator pins as GPIOs . . . . .	145
8.3.14	Using the GPIO pins in the RTC supply domain . . . . .	145
8.4	GPIO registers . . . . .	146
8.4.1	GPIO port mode register (GPIOx_MODER) (x = A..F) . . . . .	146
8.4.2	GPIO port output type register (GPIOx_OTYPER) (x = A..F) . . . . .	146
8.4.3	GPIO port output speed register (GPIOx_OSPEEDR) (x = A..F) . . . . .	147
8.4.4	GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..F) . . . . .	148
8.4.5	GPIO port input data register (GPIOx_IDR) (x = A..F) . . . . .	148
8.4.6	GPIO port output data register (GPIOx_ODR) (x = A..F) . . . . .	149
8.4.7	GPIO port bit set/reset register (GPIOx_BSRR) (x = A..F) . . . . .	149
8.4.8	GPIO port configuration lock register (GPIOx_LCKR) (x = A, B, and D) . . . . .	149

8.4.9	GPIO alternate function low register (GPIOx_AFRL) (x = A..E)	151
8.4.10	GPIO alternate function high register (GPIOx_AFRH) (x = A..F)	151
8.4.11	GPIO port bit reset register (GPIOx_BRR) (x = A..F)	152
8.4.12	GPIO register map	153
<b>9</b>	<b>System configuration controller (SYSCFG)</b>	<b>155</b>
9.1	SYSCFG registers	155
9.1.1	SYSCFG configuration register 1 (SYSCFG_CFGR1)	155
9.1.2	SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)	157
9.1.3	SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)	158
9.1.4	SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)	158
9.1.5	SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)	159
9.1.6	SYSCFG configuration register 2 (SYSCFG_CFGR2)	159
9.1.7	SYSCFG register maps	161
<b>10</b>	<b>Direct memory access controller (DMA)</b>	<b>162</b>
10.1	Introduction	162
10.2	DMA main features	162
10.3	DMA implementation	163
10.4	DMA functional description	164
10.4.1	DMA transactions	164
10.4.2	Arbiter	164
10.4.3	DMA channels	165
10.4.4	Programmable data width, data alignment and endians	166
10.4.5	Error management	167
10.4.6	DMA interrupts	168
10.4.7	DMA request mapping	169
10.5	DMA registers	174
10.5.1	DMA interrupt status register (DMA_ISR)	174
10.5.2	DMA interrupt flag clear register (DMA_IFCR)	175
10.5.3	DMA channel x configuration register (DMA_CCRx) (x = 1..7, where x= channel number)	176
10.5.4	DMA channel x number of data register (DMA_CNDTRx) (x = 1..7, where x= channel number)	178

10.5.5	DMA channel x peripheral address register (DMA_CPARx) (x = 1..7, where x = channel number) . . . . .	178
10.5.6	DMA channel x memory address register (DMA_CMARx) (x = 1..7, where x = channel number) . . . . .	179
10.5.7	DMA register map . . . . .	180
<b>11</b>	<b>Interrupts and events . . . . .</b>	<b>182</b>
11.1	Nested vectored interrupt controller (NVIC) . . . . .	182
11.1.1	NVIC main features . . . . .	182
11.1.2	SysTick calibration value register . . . . .	182
11.1.3	Interrupt and exception vectors . . . . .	182
11.2	Extended interrupts and events controller (EXTI) . . . . .	185
11.2.1	Main features . . . . .	186
11.2.2	Block diagram . . . . .	186
11.2.3	Wakeup event management . . . . .	187
11.2.4	Asynchronous Internal Interrupts . . . . .	187
11.2.5	Functional description . . . . .	188
11.2.6	External and internal interrupt/event line mapping . . . . .	189
11.3	EXTI registers . . . . .	191
11.3.1	Interrupt mask register (EXTI_IMR) . . . . .	191
11.3.2	Event mask register (EXTI_EMR) . . . . .	191
11.3.3	Rising trigger selection register (EXTI_RTSR) . . . . .	192
11.3.4	Falling trigger selection register (EXTI_FTSR) . . . . .	192
11.3.5	Software interrupt event register (EXTI_SWIER) . . . . .	193
11.3.6	Pending register (EXTI_PR) . . . . .	193
11.3.7	EXTI register map . . . . .	195
<b>12</b>	<b>Analog-to-digital converter (ADC) . . . . .</b>	<b>196</b>
12.1	ADC introduction . . . . .	196
12.2	ADC main features . . . . .	196
12.3	ADC functional description . . . . .	197
12.3.1	ADC on-off control . . . . .	198
12.3.2	ADC clock . . . . .	198
12.3.3	Channel selection . . . . .	198
12.3.4	Single conversion mode . . . . .	199
12.3.5	Continuous conversion mode . . . . .	199
12.3.6	Timing diagram . . . . .	199

12.3.7	Analog watchdog	200
12.3.8	Scan mode	201
12.3.9	Injected channel management	201
12.3.10	Discontinuous mode	202
12.4	Calibration	203
12.5	Data alignment	203
12.6	Channel-by-channel programmable sample time	204
12.7	Conversion on external trigger	204
12.8	DMA request	205
12.9	Temperature sensor and internal reference voltage	205
12.10	Battery voltage monitoring	207
12.11	ADC interrupts	207
12.12	ADC registers	208
12.12.1	ADC status register (ADC_SR)	208
12.12.2	ADC control register 1 (ADC_CR1)	209
12.12.3	ADC control register 2 (ADC_CR2)	211
12.12.4	ADC sample time register 1 (ADC_SMPR1)	213
12.12.5	ADC sample time register 2 (ADC_SMPR2)	214
12.12.6	ADC injected channel data offset register x (ADC_JOFRx)(x=1..4)	214
12.12.7	ADC watchdog high threshold register (ADC_HTR)	215
12.12.8	ADC watchdog low threshold register (ADC_LTR)	215
12.12.9	ADC regular sequence register 1 (ADC_SQR1)	216
12.12.10	ADC regular sequence register 2 (ADC_SQR2)	217
12.12.11	ADC regular sequence register 3 (ADC_SQR3)	218
12.12.12	ADC injected sequence register (ADC_JSQR)	219
12.12.13	ADC injected data register x (ADC_JDRx) (x= 1..4)	220
12.12.14	ADC regular data register (ADC_DR)	220
12.12.15	ADC register map	221
<b>13</b>	<b>Sigma-delta analog-to-digital converter (SDADC)</b>	<b>223</b>
13.1	Introduction	223
13.2	SDADC main features	224
13.3	SDADC pins	225
13.4	SDADC clock	225
13.5	SDADC functional description	226

13.5.1	SDADC on-off control	226
13.5.2	Power down and Standby low-power modes	227
13.5.3	SDADC clock	227
13.5.4	Channel selection	227
13.5.5	Differential and single-ended modes	228
13.5.6	Configuring the analog inputs	232
13.5.7	Launching calibration and determining the offset values	232
13.5.8	Launching conversions	233
13.5.9	Continuous and fast continuous modes	233
13.5.10	Request precedence	234
13.5.11	Launching conversions with deterministic timing	235
13.5.12	Reference voltage	235
13.5.13	Analog input signal ranges	236
13.5.14	Input impedance of SDADC analog input and VREFSD reference voltage	236
13.6	SDADC registers	239
13.6.1	Register write protection	239
13.6.2	SDADC control register 1 (SDADC_CR1)	239
13.6.3	SDADC control register 2 (SDADC_CR2)	242
13.6.4	SDADC interrupt and status register (SDADC_ISR)	245
13.6.5	SDADC interrupt and status clear register (SDADC_CLRISR)	247
13.6.6	SDADC injected channel group selection register (SDADC_JCHGR)	248
13.6.7	SDADC configuration 0 register (SDADC_CONF0R)	249
13.6.8	SDADC configuration 1 register (SDADC_CONF1R)	250
13.6.9	SDADC configuration 2 register (SDADC_CONF2R)	251
13.6.10	SDADC channel configuration register 1 (SDADC_CONFCHR1)	252
13.6.11	SDADC channel configuration register 2 (SDADC_CONFCHR2)	252
13.6.12	SDADC data register for injected group (SDADC_JDATAR)	253
13.6.13	SDADC data register for the regular channel (SDADC_RDATAR)	254
13.6.14	SDADC1 and SDADC2 injected data register (SDADC_JDATA12R)	255
13.6.15	SDADC1 and SDADC2 regular data register (SDADC_RDATA12R)	256
13.6.16	SDADC1 and SDADC3 injected data register (SDADC_JDATA13R)	257
13.6.17	SDADC1 and SDADC3 regular data register (SDADC_RDATA13R)	258
13.6.18	SDADC register map	259
<b>14</b>	<b>Digital-to-analog converter (DAC1 and DAC2)</b>	<b>261</b>
14.1	Introduction	261

- 14.2 DAC1/2 main features ..... 261
- 14.3 Single mode functional description ..... 264
  - 14.3.1 DAC channel enable ..... 264
  - 14.3.2 DAC output buffer enable ..... 264
  - 14.3.3 DAC data format ..... 264
  - 14.3.4 DAC channel conversion ..... 264
  - 14.3.5 DAC output voltage ..... 266
  - 14.3.6 DAC trigger selection ..... 266
- 14.4 Dual mode functional description ..... 267
  - 14.4.1 DAC channel enable ..... 267
  - 14.4.2 DAC output buffer enable ..... 267
  - 14.4.3 DAC data format ..... 267
  - 14.4.4 DAC channel conversion in dual mode ..... 268
  - 14.4.5 Description of dual conversion modes ..... 268
  - 14.4.6 DAC output voltage ..... 272
  - 14.4.7 DAC trigger selection ..... 272
- 14.5 Noise generation ..... 273
- 14.6 Triangle-wave generation ..... 274
- 14.7 DMA request ..... 275
- 14.8 DAC registers ..... 276
  - 14.8.1 DAC control register (DAC\_CR) ..... 276
  - 14.8.2 DAC software trigger register (DAC\_SWTRIGR) ..... 279
  - 14.8.3 DAC channel1 12-bit right-aligned data holding register (DAC\_DHR12R1) ..... 279
  - 14.8.4 DAC channel1 12-bit left-aligned data holding register (DAC\_DHR12L1) ..... 280
  - 14.8.5 DAC channel1 8-bit right-aligned data holding register (DAC\_DHR8R1) ..... 280
  - 14.8.6 DAC channel2 12-bit right-aligned data holding register (DAC\_DHR12R2) ..... 280
  - 14.8.7 DAC channel2 12-bit left-aligned data holding register (DAC\_DHR12L2) ..... 281
  - 14.8.8 DAC channel2 8-bit right-aligned data holding register (DAC\_DHR8R2) ..... 281
  - 14.8.9 Dual DAC 12-bit right-aligned data holding register (DAC\_DHR12RD) ..... 282
  - 14.8.10 Dual DAC 12-bit left-aligned data holding register (DAC\_DHR12LD) ..... 282

14.8.11	Dual DAC 8-bit right-aligned data holding register (DAC_DHR8RD)	283
14.8.12	DAC channel1 data output register (DAC_DOR1)	283
14.8.13	DAC channel2 data output register (DAC_DOR2)	283
14.8.14	DAC status register (DAC_SR)	284
14.8.15	DAC register map	285
<b>15</b>	<b>Comparator (COMP)</b>	<b>287</b>
15.1	Introduction	287
15.2	COMP main features	287
15.3	COMP functional description	288
15.3.1	COMP block diagram	288
15.3.2	COMP pins and internal signals	288
15.3.3	COMP reset and clocks	289
15.3.4	Comparator LOCK mechanism	289
15.3.5	Hysteresis	289
15.4	Power mode	290
15.5	Interrupts	290
15.6	COMP registers	291
15.6.1	COMP control and status register (COMP_CSR)	291
15.6.2	COMP register map	294
<b>16</b>	<b>General-purpose timers (TIM2 to TIM5, TIM19)</b>	<b>295</b>
16.1	TIM2 to TIM5/TIM19 introduction	295
16.2	TIM2 to TIM5/TIM19 main features	295
16.3	TIM2 to TIM5/TIM19 functional description	296
16.3.1	Time-base unit	296
16.3.2	Counter modes	298
16.3.3	Clock selection	309
16.3.4	Capture/compare channels	312
16.3.5	Input capture mode	314
16.3.6	PWM input mode	315
16.3.7	Forced output mode	316
16.3.8	Output compare mode	317
16.3.9	PWM mode	318
16.3.10	One-pulse mode	321

16.3.11	Clearing the OCxREF signal on an external event	322
16.3.12	Encoder interface mode	323
16.3.13	Timer input XOR function	325
16.3.14	Timers and external trigger synchronization	326
16.3.15	Timer synchronization	329
16.3.16	Debug mode	334
<b>16.4</b>	<b>TIM2 to TIM5/TIM19 registers</b>	<b>335</b>
16.4.1	TIMx control register 1 (TIMx_CR1)	335
16.4.2	TIMx control register 2 (TIMx_CR2)	337
16.4.3	TIMx slave mode control register (TIMx_SMCR)	338
16.4.4	TIMx DMA/Interrupt enable register (TIMx_DIER)	340
16.4.5	TIMx status register (TIMx_SR)	341
16.4.6	TIMx event generation register (TIMx_EGR)	343
16.4.7	TIMx capture/compare mode register 1 (TIMx_CCMR1)	344
16.4.8	TIMx capture/compare mode register 2 (TIMx_CCMR2)	347
16.4.9	TIMx capture/compare enable register (TIMx_CCER)	348
16.4.10	TIMx counter (TIMx_CNT)	350
16.4.11	TIMx prescaler (TIMx_PSC)	350
16.4.12	TIMx auto-reload register (TIMx_ARR)	350
16.4.13	TIMx capture/compare register 1 (TIMx_CCR1)	351
16.4.14	TIMx capture/compare register 2 (TIMx_CCR2)	351
16.4.15	TIMx capture/compare register 3 (TIMx_CCR3)	352
16.4.16	TIMx capture/compare register 4 (TIMx_CCR4)	352
16.4.17	TIMx DMA control register (TIMx_DCR)	353
16.4.18	TIMx DMA address for full transfer (TIMx_DMAR)	353
16.4.19	TIM2 option register (TIM2_OR)	355
16.4.20	TIMx register map	355
<b>17</b>	<b>General-purpose timers (TIM12/13/14)</b>	<b>358</b>
17.1	TIM12/13/14 introduction	358
17.2	TIM12/13/14 main features	358
17.2.1	TIM12 main features	358
17.3	TIM13/TIM14 main features	360
17.4	TIM12/13/14 functional description	361
17.4.1	Time-base unit	361
17.4.2	Counter modes	363
17.4.3	Clock selection	367

17.4.4	Capture/compare channels	369
17.4.5	Input capture mode	371
17.4.6	PWM input mode (only for TIM12)	372
17.4.7	Forced output mode	373
17.4.8	Output compare mode	373
17.4.9	PWM mode	374
17.4.10	One-pulse mode (only for TIM12)	376
17.4.11	TIM12 external trigger synchronization	377
17.4.12	Timer synchronization (TIM12)	380
17.4.13	Debug mode	380
17.5	TIM12 registers	381
17.5.1	TIM12 control register 1 (TIMx_CR1)	381
17.5.2	TIM12 slave mode control register (TIMx_SMCR)	382
17.5.3	TIM12 Interrupt enable register (TIMx_DIER)	383
17.5.4	TIM12 status register (TIMx_SR)	384
17.5.5	TIM12 event generation register (TIMx_EGR)	386
17.5.6	TIM12 capture/compare mode register 1 (TIMx_CCMR1)	387
17.5.7	TIM12 capture/compare enable register (TIMx_CCER)	390
17.5.8	TIM12 counter (TIMx_CNT)	391
17.5.9	TIM12 prescaler (TIMx_PSC)	391
17.5.10	TIM12 auto-reload register (TIMx_ARR)	391
17.5.11	TIM12 capture/compare register 1 (TIMx_CCR1)	392
17.5.12	TIM12 capture/compare register 2 (TIMx_CCR2)	392
17.5.13	TIM12 register map	393
17.6	TIM13/14 registers	395
17.6.1	TIM13/14 control register 1 (TIMx_CR1)	395
17.6.2	TIM13/14 Interrupt enable register (TIMx_DIER)	396
17.6.3	TIM13/14 status register (TIMx_SR)	396
17.6.4	TIM13/14 event generation register (TIMx_EGR)	397
17.6.5	TIM13/14 capture/compare mode register 1 (TIMx_CCMR1)	398
17.6.6	TIM13/14 capture/compare enable register (TIMx_CCER)	401
17.6.7	TIM13/14 counter (TIMx_CNT)	402
17.6.8	TIM13/14 prescaler (TIMx_PSC)	402
17.6.9	TIM13/14 auto-reload register (TIMx_ARR)	402
17.6.10	TIM13/14 capture/compare register 1 (TIMx_CCR1)	403

17.6.11	TIM14 option register (TIM14_OR)	404
17.6.12	TIM13/14 register map	404
<b>18</b>	<b>General-purpose timers (TIM15/16/17)</b>	<b>406</b>
18.1	TIM15/16/17 introduction	406
18.2	TIM15 main features	406
18.3	TIM16 and TIM17 main features	407
18.4	TIM15/16/17 functional description	410
18.4.1	Time-base unit	410
18.4.2	Counter modes	412
18.4.3	Repetition counter	416
18.4.4	Clock selection	417
18.4.5	Capture/compare channels	419
18.4.6	Input capture mode	422
18.4.7	PWM input mode (only for TIM15)	423
18.4.8	Forced output mode	423
18.4.9	Output compare mode	424
18.4.10	PWM mode	425
18.4.11	Complementary outputs and dead-time insertion	426
18.4.12	Using the break function	429
18.4.13	One-pulse mode	432
18.4.14	TIM15 and external trigger synchronization (only for TIM15)	434
18.4.15	Timer synchronization	436
18.4.16	Debug mode	436
18.5	TIM15 registers	437
18.5.1	TIM15 control register 1 (TIM15_CR1)	437
18.5.2	TIM15 control register 2 (TIM15_CR2)	438
18.5.3	TIM15 slave mode control register (TIM15_SMCR)	440
18.5.4	TIM15 DMA/interrupt enable register (TIM15_DIER)	442
18.5.5	TIM15 status register (TIM15_SR)	443
18.5.6	TIM15 event generation register (TIM15_EGR)	444
18.5.7	TIM15 capture/compare mode register 1 (TIM15_CCMR1)	445
18.5.8	TIM15 capture/compare enable register (TIM15_CCER)	448
18.5.9	TIM15 counter (TIM15_CNT)	451
18.5.10	TIM15 prescaler (TIM15_PSC)	451
18.5.11	TIM15 auto-reload register (TIM15_ARR)	451
18.5.12	TIM15 repetition counter register (TIM15_RCR)	452

18.5.13	TIM15 capture/compare register 1 (TIM15_CCR1) . . . . .	452
18.5.14	TIM15 capture/compare register 2 (TIM15_CCR2) . . . . .	453
18.5.15	TIM15 break and dead-time register (TIM15_BDTR) . . . . .	453
18.5.16	TIM15 DMA control register (TIM15_DCR) . . . . .	455
18.5.17	TIM15 DMA address for full transfer (TIM15_DMAR) . . . . .	456
18.5.18	TIM15 register map . . . . .	457
<b>18.6</b>	<b>TIM16&amp;TIM17 registers . . . . .</b>	<b>459</b>
18.6.1	TIM16&TIM17 control register 1 (TIMx_CR1) . . . . .	459
18.6.2	TIM16&TIM17 control register 2 (TIMx_CR2) . . . . .	460
18.6.3	TIM16&TIM17 DMA/interrupt enable register (TIMx_DIER) . . . . .	462
18.6.4	TIM16&TIM17 status register (TIMx_SR) . . . . .	463
18.6.5	TIM16&TIM17 event generation register (TIMx_EGR) . . . . .	464
18.6.6	TIM16&TIM17 capture/compare mode register 1 (TIMx_CCMR1) . . .	465
18.6.7	TIM16&TIM17 capture/compare enable register (TIMx_CCER) . . . . .	468
18.6.8	TIM16&TIM17 counter (TIMx_CNT) . . . . .	471
18.6.9	TIM16&TIM17 prescaler (TIMx_PSC) . . . . .	471
18.6.10	TIM16&TIM17 auto-reload register (TIMx_ARR) . . . . .	471
18.6.11	TIM16&TIM17 repetition counter register (TIMx_RCR) . . . . .	472
18.6.12	TIM16&TIM17 capture/compare register 1 (TIMx_CCR1) . . . . .	472
18.6.13	TIM16&TIM17 break and dead-time register (TIMx_BDTR) . . . . .	473
18.6.14	TIM16&TIM17 DMA control register (TIMx_DCR) . . . . .	474
18.6.15	TIM16&TIM17 DMA address for full transfer (TIMx_DMAR) . . . . .	475
18.6.16	TIM16&TIM17 register map . . . . .	476
<b>19</b>	<b>Infrared interface (IRTIM) . . . . .</b>	<b>478</b>
<b>20</b>	<b>Basic timers (TIM6/7/18) . . . . .</b>	<b>479</b>
20.1	Introduction . . . . .	479
20.2	TIM6/7/18 main features . . . . .	479
20.3	TIM6/7/18 functional description . . . . .	480
20.3.1	Time-base unit . . . . .	480
20.3.2	Counting mode . . . . .	482
20.3.3	Clock source . . . . .	485
20.3.4	Debug mode . . . . .	486
20.4	TIM6/7/18 registers . . . . .	486
20.4.1	TIM6/7/18 control register 1 (TIMx_CR1) . . . . .	486
20.4.2	TIM6/7/18 control register 2 (TIMx_CR2) . . . . .	488

20.4.3	TIM6/7/18 DMA/Interrupt enable register (TIMx_DIER) . . . . .	488
20.4.4	TIM6/7/18 status register (TIMx_SR) . . . . .	489
20.4.5	TIM6/7/18 event generation register (TIMx_EGR) . . . . .	489
20.4.6	TIM6/7/18 counter (TIMx_CNT) . . . . .	489
20.4.7	TIM6/7/18 prescaler (TIMx_PSC) . . . . .	490
20.4.8	TIM6/7/18 auto-reload register (TIMx_ARR) . . . . .	490
20.4.9	TIM6/7/18 register map . . . . .	491
<b>21</b>	<b>Independent watchdog (IWDG) . . . . .</b>	<b>492</b>
21.1	Introduction . . . . .	492
21.2	IWDG main features . . . . .	492
21.3	IWDG functional description . . . . .	492
21.3.1	IWDG block diagram . . . . .	492
21.3.2	Window option . . . . .	493
21.3.3	Hardware watchdog . . . . .	493
21.3.4	Register access protection . . . . .	494
21.3.5	Debug mode . . . . .	494
21.4	IWDG registers . . . . .	494
21.4.1	Key register (IWDG_KR) . . . . .	494
21.4.2	Prescaler register (IWDG_PR) . . . . .	495
21.4.3	Reload register (IWDG_RLR) . . . . .	496
21.4.4	Status register (IWDG_SR) . . . . .	497
21.4.5	Window register (IWDG_WINR) . . . . .	498
21.4.6	IWDG register map . . . . .	499
<b>22</b>	<b>System window watchdog (WWDG) . . . . .</b>	<b>500</b>
22.1	Introduction . . . . .	500
22.2	WWDG main features . . . . .	500
22.3	WWDG functional description . . . . .	500
22.3.1	Enabling the watchdog . . . . .	501
22.3.2	Controlling the downcounter . . . . .	501
22.3.3	Advanced watchdog interrupt feature . . . . .	501
22.3.4	How to program the watchdog timeout . . . . .	502
22.3.5	Debug mode . . . . .	503
22.4	WWDG registers . . . . .	503
22.4.1	Control register (WWDG_CR) . . . . .	503

22.4.2	Configuration register (WWDG_CFR) .....	504
22.4.3	Status register (WWDG_SR) .....	504
22.4.4	WWDG register map .....	505
<b>23</b>	<b>Real-time clock (RTC) .....</b>	<b>506</b>
23.1	Introduction .....	506
23.2	RTC main features .....	507
23.3	RTC functional description .....	508
23.3.1	RTC block diagram .....	508
23.3.2	GPIOs controlled by the RTC .....	509
23.3.3	Clock and prescalers .....	510
23.3.4	Real-time clock and calendar .....	511
23.3.5	Programmable alarms .....	512
23.3.6	Periodic auto-wakeup .....	512
23.3.7	RTC initialization and configuration .....	513
23.3.8	Reading the calendar .....	514
23.3.9	Resetting the RTC .....	515
23.3.10	RTC synchronization .....	516
23.3.11	RTC reference clock detection .....	516
23.3.12	RTC smooth digital calibration .....	517
23.3.13	Time-stamp function .....	519
23.3.14	Tamper detection .....	520
23.3.15	Calibration clock output .....	521
23.3.16	Alarm output .....	522
23.4	RTC low-power modes .....	522
23.5	RTC interrupts .....	522
23.6	RTC registers .....	523
23.6.1	RTC time register (RTC_TR) .....	524
23.6.2	RTC date register (RTC_DR) .....	525
23.6.3	RTC control register (RTC_CR) .....	526
23.6.4	RTC initialization and status register (RTC_ISR) .....	529
23.6.5	RTC prescaler register (RTC_PRER) .....	532
23.6.6	RTC wakeup timer register (RTC_WUTR) .....	533
23.6.7	RTC alarm A register (RTC_ALRMAR) .....	534
23.6.8	RTC alarm B register (RTC_ALRMBR) .....	535
23.6.9	RTC write protection register (RTC_WPR) .....	536

23.6.10	RTC sub second register (RTC_SSR)	537
23.6.11	RTC shift control register (RTC_SHIFTR)	538
23.6.12	RTC timestamp time register (RTC_TSTR)	539
23.6.13	RTC timestamp date register (RTC_TSDR)	540
23.6.14	RTC time-stamp sub second register (RTC_TSSSR)	541
23.6.15	RTC calibration register (RTC_CALR)	542
23.6.16	RTC tamper and alternate function configuration register (RTC_TAFCR)	543
23.6.17	RTC alarm A sub second register (RTC_ALRMASR)	546
23.6.18	RTC alarm B sub second register (RTC_ALRMBSSR)	547
23.6.19	RTC backup registers (RTC_BKPxR)	548
23.6.20	RTC register map	549
<b>24</b>	<b>Inter-integrated circuit (I2C) interface</b>	<b>551</b>
24.1	Introduction	551
24.2	I2C main features	551
24.3	I2C implementation	552
24.4	I2C functional description	552
24.4.1	I2C block diagram	553
24.4.2	I2C clock requirements	554
24.4.3	Mode selection	554
24.4.4	I2C initialization	556
24.4.5	Software reset	560
24.4.6	Data transfer	561
24.4.7	I2C slave mode	563
24.4.8	I2C master mode	572
24.4.9	I2Cx_TIMINGR register configuration examples	584
24.4.10	SMBus specific features	585
24.4.11	SMBus initialization	588
24.4.12	SMBus: I2Cx_TIMEOCTR register configuration examples	590
24.4.13	SMBus slave mode	591
24.4.14	Wakeup from Stop mode on address match	598
24.4.15	Error conditions	599
24.4.16	DMA requests	601
24.4.17	Debug mode	602
24.5	I2C low-power modes	602
24.6	I2C interrupts	602

24.7	I2C registers	603
24.7.1	Control register 1 (I2Cx_CR1)	604
24.7.2	Control register 2 (I2Cx_CR2)	607
24.7.3	Own address 1 register (I2Cx_OAR1)	610
24.7.4	Own address 2 register (I2Cx_OAR2)	611
24.7.5	Timing register (I2Cx_TIMINGR)	612
24.7.6	Timeout register (I2Cx_TIMEOUTR)	613
24.7.7	Interrupt and Status register (I2Cx_ISR)	614
24.7.8	Interrupt clear register (I2Cx_ICR)	616
24.7.9	PEC register (I2Cx_PECR)	617
24.7.10	Receive data register (I2Cx_RXDR)	618
24.7.11	Transmit data register (I2Cx_TXDR)	618
24.7.12	I2C register map	619
<b>25</b>	<b>Universal synchronous asynchronous receiver transmitter (USART)</b>	<b>621</b>
25.1	USART introduction	621
25.2	USART main features	621
25.3	USART extended features	622
25.4	USART implementation	623
25.5	USART functional description	623
25.5.1	USART character description	626
25.5.2	Transmitter	627
25.5.3	Receiver	629
25.5.4	Baud rate generation	635
25.5.5	Tolerance of the USART receiver to clock deviation	637
25.5.6	Auto baud rate detection	638
25.5.7	Multiprocessor communication	639
25.5.8	ModBus communication	641
25.5.9	Parity control	641
25.5.10	LIN (local interconnection network) mode	642
25.5.11	USART synchronous mode	645
25.5.12	Single-wire half-duplex communication	647
25.5.13	Smartcard mode	648
25.5.14	IrDA SIR ENDEC block	652
25.5.15	Continuous communication using DMA	655
25.5.16	Hardware flow control and RS485 Driver Enable	657

25.5.17	Wakeup from Stop mode	659
25.6	USART interrupts	660
25.7	USART registers	662
25.7.1	Control register 1 (USART_CR1)	662
25.7.2	Control register 2 (USART_CR2)	666
25.7.3	Control register 3 (USART_CR3)	670
25.7.4	Baud rate register (USART_BRR)	673
25.7.5	Guard time and prescaler register (USART_GTPR)	673
25.7.6	Receiver timeout register (USART_RTOR)	674
25.7.7	Request register (USART_RQR)	676
25.7.8	Interrupt & status register (USART_ISR)	677
25.7.9	Interrupt flag clear register (USART_ICR)	681
25.7.10	Receive data register (USART_RDR)	683
25.7.11	Transmit data register (USART_TDR)	684
25.7.12	USART register map	685
<b>26</b>	<b>Serial peripheral interface / inter-IC sound (SPI/I2S)</b>	<b>686</b>
26.1	Introduction	686
26.2	SPI main features	686
26.3	I2S main features	687
26.4	SPI/I2S implementation	687
26.5	SPI functional description	688
26.5.1	General description	688
26.5.2	Communications between one master and one slave	689
26.5.3	Standard multi-slave communication	691
26.5.4	Slave select (NSS) pin management	692
26.5.5	Communication formats	694
26.5.6	Configuration of SPI	696
26.5.7	Procedure for enabling SPI	697
26.5.8	Data transmission and reception procedures	697
26.5.9	SPI status flags	707
26.5.10	SPI error flags	708
26.5.11	NSS pulse mode	709
26.5.12	TI mode	709
26.5.13	CRC calculation	710
26.6	SPI interrupts	712

26.7	I <sup>2</sup> S functional description	713
26.7.1	I <sup>2</sup> S general description	713
26.7.2	Supported audio protocols	714
26.7.3	Clock generator	720
26.7.4	I <sup>2</sup> S master mode	723
26.7.5	I <sup>2</sup> S slave mode	725
26.7.6	I <sup>2</sup> S status flags	727
26.7.7	I <sup>2</sup> S error flags	728
26.7.8	DMA features	728
26.8	I <sup>2</sup> S interrupts	729
26.9	SPI and I <sup>2</sup> S registers	730
26.9.1	SPI control register 1 (SPIx_CR1)	730
26.9.2	SPI control register 2 (SPIx_CR2)	732
26.9.3	SPI status register (SPIx_SR)	735
26.9.4	SPI data register (SPIx_DR)	737
26.9.5	SPI CRC polynomial register (SPIx_CRCPR)	737
26.9.6	SPI Rx CRC register (SPIx_RXCR)	738
26.9.7	SPI Tx CRC register (SPIx_TXCR)	738
26.9.8	SPIx_I <sup>2</sup> S configuration register (SPIx_I2SCFGR)	739
26.9.9	SPIx_I <sup>2</sup> S prescaler register (SPIx_I2SPR)	741
26.9.10	SPI/I <sup>2</sup> S register map	742
<b>27</b>	<b>Touch sensing controller (TSC)</b>	<b>743</b>
27.1	Introduction	743
27.2	TSC main features	743
27.3	TSC functional description	744
27.3.1	TSC block diagram	744
27.3.2	Surface charge transfer acquisition overview	744
27.3.3	Reset and clocks	746
27.3.4	Charge transfer acquisition sequence	747
27.3.5	Spread spectrum feature	748
27.3.6	Max count error	748
27.3.7	Sampling capacitor I/O and channel I/O mode selection	749
27.3.8	Acquisition mode	750
27.3.9	I/O hysteresis and analog switch control	750
27.3.10	Capacitive sensing GPIOs	751

27.4	TSC low-power modes	751
27.5	TSC interrupts	752
27.6	TSC registers	752
27.6.1	TSC control register (TSC_CR)	752
27.6.2	TSC interrupt enable register (TSC_IER)	754
27.6.3	TSC interrupt clear register (TSC_ICR)	755
27.6.4	TSC interrupt status register (TSC_ISR)	756
27.6.5	TSC I/O hysteresis control register (TSC_IOHCR)	756
27.6.6	TSC I/O analog switch control register (TSC_IOASCR)	757
27.6.7	TSC I/O sampling control register (TSC_IOSCR)	758
27.6.8	TSC I/O channel control register (TSC_IOCOCR)	758
27.6.9	TSC I/O group control status register (TSC_ILOGCSR)	759
27.6.10	TSC I/O group x counter register (TSC_ILOGxCR) (x = 1..8)	759
27.6.11	TSC register map	760
<b>28</b>	<b>Controller area network (bxCAN)</b>	<b>762</b>
28.1	Introduction	762
28.2	bxCAN main features	762
28.3	bxCAN general description	763
28.3.1	CAN 2.0B active core	763
28.3.2	Control, status and configuration registers	763
28.3.3	Tx mailboxes	763
28.3.4	Acceptance filters	764
28.4	bxCAN operating modes	764
28.4.1	Initialization mode	764
28.4.2	Normal mode	764
28.4.3	Sleep mode (low-power)	765
28.5	Test mode	766
28.5.1	Silent mode	766
28.5.2	Loop back mode	767
28.5.3	Loop back combined with silent mode	767
28.6	Behavior in Debug mode	768
28.7	bxCAN functional description	768
28.7.1	Transmission handling	768
28.7.2	Time triggered communication mode	770
28.7.3	Reception handling	770

	28.7.4	Identifier filtering	771
	28.7.5	Message storage	775
	28.7.6	Error management	777
	28.7.7	Bit timing	777
28.8		bxCAN interrupts	780
28.9		CAN registers	781
	28.9.1	Register access protection	781
	28.9.2	CAN control and status registers	781
	28.9.3	CAN mailbox registers	792
	28.9.4	CAN filter registers	799
	28.9.5	bxCAN register map	803
<b>29</b>		<b>Universal serial bus full-speed device interface (USB)</b>	<b>807</b>
	29.1	Introduction	807
	29.2	USB main features	807
	29.3	USB implementation	807
	29.4	USB functional description	808
	29.4.1	Description of USB blocks	809
	29.5	Programming considerations	810
	29.5.1	Generic USB device programming	810
	29.5.2	System and power-on reset	811
	29.5.3	Double-buffered endpoints	815
	29.5.4	Isochronous transfers	818
	29.5.5	Suspend/Resume events	819
	29.6	USB registers	821
	29.6.1	Common registers	821
	29.6.2	Endpoint-specific registers	827
	29.6.3	Buffer descriptor table	832
	29.6.4	USB register map	835
<b>30</b>		<b>HDMI-CEC controller (HDMI-CEC)</b>	<b>837</b>
	30.1	Introduction	837
	30.2	HDMI-CEC controller main features	837
	30.3	HDMI-CEC functional description	838
	30.3.1	HDMI-CEC pin	838
	30.3.2	Message description	839

30.3.3	Bit timing	839
30.4	Arbitration	840
30.4.1	SFT option bit	841
30.5	Error handling	842
30.5.1	Bit error	842
30.5.2	Message error	842
30.5.3	Bit Rising Error (BRE)	843
30.5.4	Short Bit Period Error (SBPE)	843
30.5.5	Long Bit Period Error (LBPE)	843
30.5.6	Transmission Error Detection (TXERR)	845
30.6	HDMI-CEC interrupts	846
30.7	HDMI-CEC registers	847
30.7.1	CEC control register (CEC_CR)	847
30.7.2	CEC configuration register (CEC_CFGR)	848
30.7.3	CEC Tx data register (CEC_TXDR)	851
30.7.4	CEC Rx Data Register (CEC_RXDR)	851
30.7.5	CEC Interrupt and Status Register (CEC_ISR)	851
30.7.6	CEC interrupt enable register (CEC_IER)	853
30.7.7	HDMI-CEC register map	855
<b>31</b>	<b>Debug support (DBG)</b>	<b>856</b>
31.1	Overview	856
31.2	Reference ARM documentation	857
31.3	SWJ debug port (serial wire and JTAG)	857
31.3.1	Mechanism to select the JTAG-DP or the SW-DP	858
31.4	Pinout and debug port pins	858
31.4.1	SWJ debug port pins	859
31.4.2	Flexible SWJ-DP pin assignment	859
31.4.3	Internal pull-up and pull-down on JTAG pins	860
31.4.4	Using serial wire and releasing the unused debug pins as GPIOs	861
31.5	STM32F37xxx JTAG TAP connection	861
31.6	ID codes and locking mechanism	862
31.6.1	MCU device ID code	863
31.6.2	Boundary scan TAP	863
31.6.3	Cortex <sup>®</sup> -M4 with FPU TAP	863
31.6.4	Cortex <sup>®</sup> -M4 with FPU JEDEC-106 ID code	864

31.7	JTAG debug port	864
31.8	SW debug port	866
31.8.1	SW protocol introduction	866
31.8.2	SW protocol sequence	866
31.8.3	SW-DP state machine (reset, idle states, ID code)	867
31.8.4	DP and AP read/write accesses	867
31.8.5	SW-DP registers	868
31.8.6	SW-AP registers	868
31.9	AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP	869
31.10	Core debug	870
31.11	Capability of the debugger host to connect under system reset	870
31.12	FPB (Flash patch breakpoint)	871
31.13	DWT (data watchpoint trigger)	871
31.14	ITM (instrumentation trace macrocell)	871
31.14.1	General description	871
31.14.2	Time stamp packets, synchronization and overflow packets	872
31.15	ETM (Embedded trace macrocell)	873
31.15.1	General description	873
31.15.2	Signal protocol, packet types	874
31.15.3	Main ETM registers	874
31.15.4	Configuration example	874
31.16	MCU debug component (DBGMCU)	874
31.16.1	Debug support for low-power modes	875
31.16.2	Debug support for timers, watchdog, bxCAN and I <sup>2</sup> C	875
31.16.3	Debug MCU configuration register	875
31.16.4	Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)	878
31.16.5	Debug MCU APB2 freeze register (DBGMCU_APB2_FZ)	880
31.17	TPIU (trace port interface unit)	880
31.17.1	Introduction	880
31.17.2	TRACE pin assignment	881
31.17.3	TPUI formatter	883
31.17.4	TPUI frame synchronization packets	883
31.17.5	Transmission of the synchronization frame packet	883
31.17.6	Synchronous mode	884
31.17.7	Asynchronous mode	884

---

31.17.8	TRACECLKIN connection inside the STM32F37xxx	884
31.17.9	TPIU registers	885
31.17.10	Example of configuration	886
31.18	DBG register map	886
<b>32</b>	<b>Device electronic signature</b>	<b>888</b>
32.1	Unique device ID register (96 bits)	888
32.2	Memory size data register	889
32.2.1	Flash size data register	889
<b>33</b>	<b>Revision history</b>	<b>893</b>

## List of tables

Table 1.	STM32F37xx peripheral register boundary addresses . . . . .	39
Table 2.	Boot modes . . . . .	44
Table 3.	Flash module organization . . . . .	46
Table 4.	Flash memory read protection status . . . . .	55
Table 5.	Access status versus protection level and execution modes . . . . .	57
Table 6.	Flash interrupt request . . . . .	58
Table 7.	Flash interface - register map and reset values . . . . .	64
Table 8.	Option byte format . . . . .	66
Table 9.	Option byte organization . . . . .	66
Table 10.	Description of the option bytes . . . . .	67
Table 11.	CRC register map and reset values . . . . .	74
Table 12.	Low-power mode summary . . . . .	82
Table 13.	Sleep-now . . . . .	84
Table 14.	Sleep-on-exit . . . . .	84
Table 15.	Stop mode . . . . .	86
Table 16.	Standby mode . . . . .	87
Table 17.	PWR register map and reset values . . . . .	93
Table 18.	RCC register map and reset values . . . . .	135
Table 19.	Port bit configuration table . . . . .	139
Table 20.	GPIO register map and reset values . . . . .	153
Table 21.	SYSCFG register map and reset values . . . . .	161
Table 22.	DMA implementation . . . . .	163
Table 23.	Programmable data width & endian behavior (when bits PINC = MINC = 1) . . . . .	166
Table 24.	DMA interrupt requests . . . . .	168
Table 25.	Summary of DMA1 requests for each channel . . . . .	171
Table 26.	Summary of DMA2 requests for each channel . . . . .	173
Table 27.	DMA register map and reset values . . . . .	180
Table 28.	List of vectors . . . . .	182
Table 29.	Extended interrupt/event controller register map and reset values . . . . .	195
Table 30.	ADC pins . . . . .	198
Table 31.	Analog watchdog channel selection . . . . .	200
Table 32.	External trigger for regular channels for ADC1 . . . . .	205
Table 33.	External trigger for injected channels for ADC1 . . . . .	205
Table 34.	ADC interrupts . . . . .	207
Table 35.	ADC register map and reset values . . . . .	221
Table 36.	ADC pins . . . . .	225
Table 37.	SDADC register map and reset values . . . . .	259
Table 38.	DAC1/2 pins . . . . .	263
Table 39.	External triggers (DAC1) . . . . .	266
Table 40.	External triggers (DAC2) . . . . .	266
Table 41.	External triggers (DAC1) . . . . .	272
Table 42.	DAC register map and reset values . . . . .	285
Table 43.	COMP register map and reset values . . . . .	294
Table 44.	Counting direction versus encoder signals . . . . .	324
Table 45.	TIMx internal trigger connection . . . . .	340
Table 46.	Output control bit for standard OCx channels . . . . .	349
Table 47.	TIM2 to TIM15/19 register map and reset values . . . . .	355
Table 48.	TIMx Internal trigger connection . . . . .	383

Table 49.	Output control bit for standard OCx channels . . . . .	391
Table 50.	TIM12 register map and reset values . . . . .	393
Table 51.	Output control bit for standard OCx channels . . . . .	401
Table 52.	TIM13/14 register map and reset values . . . . .	404
Table 53.	TIMx Internal trigger connection . . . . .	441
Table 54.	Output control bits for complementary OCx and OCxN channels with break feature . . . . .	450
Table 55.	TIM15 register map and reset values . . . . .	457
Table 56.	Output control bits for complementary OCx and OCxN channels with break feature . . . . .	470
Table 57.	TIM16&TIM17 register map and reset values . . . . .	476
Table 58.	TIM6/7/18 register map and reset values . . . . .	491
Table 59.	IWDG register map and reset values . . . . .	499
Table 60.	WWDG register map and reset values . . . . .	505
Table 61.	RTC pin PC13 configuration . . . . .	509
Table 62.	LSE pin PC14 configuration . . . . .	510
Table 63.	LSE pin PC15 configuration . . . . .	510
Table 64.	Effect of low-power modes on RTC . . . . .	522
Table 65.	Interrupt control bits . . . . .	523
Table 66.	RTC register map and reset values . . . . .	549
Table 67.	STM32F37xx I2C implementation . . . . .	552
Table 68.	Comparison of analog vs. digital filters . . . . .	556
Table 69.	I2C-SMBUS specification data setup and hold times . . . . .	559
Table 70.	I2C Configuration table . . . . .	563
Table 71.	I2C-SMBUS specification clock timings . . . . .	573
Table 72.	Examples of timings settings for fI2CCLK = 8 MHz . . . . .	584
Table 73.	Examples of timings settings for fI2CCLK = 16 MHz . . . . .	584
Table 74.	Examples of timings settings for fI2CCLK = 48 MHz . . . . .	585
Table 75.	SMBus timeout specifications . . . . .	587
Table 76.	SMBUS with PEC configuration table . . . . .	589
Table 77.	Examples of TIMEOUTA settings for various I2CCLK frequencies (max t <sub>TIMEOUT</sub> = 25 ms) . . . . .	590
Table 78.	Examples of TIMEOUTB settings for various I2CCLK frequencies . . . . .	591
Table 79.	Examples of TIMEOUTA settings for various I2CCLK frequencies (max t <sub>IDLE</sub> = 50 μs) . . . . .	591
Table 80.	low-power modes . . . . .	602
Table 81.	I2C Interrupt requests . . . . .	602
Table 82.	I2C register map and reset values . . . . .	619
Table 83.	USART features . . . . .	623
Table 84.	Noise detection from sampled data . . . . .	634
Table 85.	Error calculation for programmed baud rates at f <sub>CK</sub> = 72 MHz for oversampling by 16 and by 8 . . . . .	636
Table 86.	Tolerance of the USART receiver when BRR[3:0] = 0000 . . . . .	637
Table 87.	Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . . .	638
Table 88.	Frame formats . . . . .	642
Table 89.	USART interrupt requests . . . . .	660
Table 90.	USART register map and reset values . . . . .	685
Table 91.	SPI interrupt requests . . . . .	712
Table 92.	Audio-frequency precision using standard 8 MHz HSE . . . . .	722
Table 93.	Audio-frequency precision using standard 8 MHz HSE . . . . .	723
Table 94.	I <sup>2</sup> S interrupt requests . . . . .	729
Table 95.	SPI register map and reset values . . . . .	742
Table 96.	Acquisition sequence summary . . . . .	746

Table 97.	Spread spectrum deviation versus AHB clock frequency . . . . .	748
Table 98.	I/O state depending on its mode and IODEF bit value . . . . .	749
Table 99.	Capacitive sensing GPIOs available on STM32F378xx devices . . . . .	751
Table 100.	Effect of low-power modes on TSC . . . . .	751
Table 101.	Interrupt control bits . . . . .	752
Table 102.	TSC register map and reset values . . . . .	760
Table 103.	Transmit mailbox mapping . . . . .	776
Table 104.	Receive mailbox mapping . . . . .	776
Table 105.	bxCAN register map and reset values . . . . .	803
Table 106.	STM32F37xx USB implementation . . . . .	807
Table 107.	Double-buffering buffer flag definition . . . . .	816
Table 108.	Bulk double-buffering memory buffers usage . . . . .	817
Table 109.	Isochronous memory buffers usage . . . . .	818
Table 110.	Resume event detection . . . . .	820
Table 111.	Reception status encoding . . . . .	830
Table 112.	Endpoint type encoding . . . . .	831
Table 113.	Endpoint kind meaning . . . . .	831
Table 114.	Transmission status encoding . . . . .	831
Table 115.	Definition of allocated buffer memory . . . . .	835
Table 116.	USB register map and reset values . . . . .	835
Table 117.	HDMI pin . . . . .	838
Table 118.	Error handling timing parameters . . . . .	844
Table 119.	TXERR timing parameters . . . . .	845
Table 120.	HDMI-CEC interrupts . . . . .	846
Table 121.	HDMI-CEC register map and reset values . . . . .	855
Table 122.	SWJ debug port pins . . . . .	859
Table 123.	Flexible SWJ-DP pin assignment . . . . .	859
Table 124.	JTAG debug port data registers . . . . .	864
Table 125.	32-bit debug port registers addressed through the shifted value A[3:2] . . . . .	865
Table 126.	Packet request (8-bits) . . . . .	866
Table 127.	ACK response (3 bits) . . . . .	867
Table 128.	DATA transfer (33 bits) . . . . .	867
Table 129.	SW-DP registers . . . . .	868
Table 130.	Cortex <sup>®</sup> -M4 with FPU AHB-AP registers . . . . .	869
Table 131.	Core debug registers . . . . .	870
Table 132.	Main ITM registers . . . . .	872
Table 133.	Main ETM registers . . . . .	874
Table 134.	Asynchronous TRACE pin assignment . . . . .	881
Table 135.	Synchronous TRACE pin assignment . . . . .	881
Table 136.	Flexible TRACE pin assignment . . . . .	882
Table 137.	Important TPIU registers . . . . .	885
Table 138.	DBG register map and reset values . . . . .	886
Table 139.	Document revision history . . . . .	893

## List of figures

Figure 1.	System architecture	37
Figure 2.	Programming procedure	50
Figure 3.	Flash memory Page Erase procedure	52
Figure 4.	Flash memory Mass Erase procedure	53
Figure 5.	CRC calculation unit block diagram	70
Figure 6.	Power supply overview	75
Figure 7.	Power on reset/power down reset waveform	80
Figure 8.	PVD thresholds	81
Figure 9.	Simplified diagram of the reset circuit	95
Figure 10.	Clock tree part 1	98
Figure 11.	Clock tree part 2	99
Figure 12.	HSE/ LSE clock sources	100
Figure 13.	Frequency measurement with TIM14 in capture mode	105
Figure 14.	Basic structure of an I/O port bit	138
Figure 15.	Basic structure of a five-volt tolerant I/O port bit	138
Figure 16.	Input floating/pull up/pull down configurations	143
Figure 17.	Output configuration	144
Figure 18.	Alternate function configuration	144
Figure 19.	High impedance-analog configuration	145
Figure 20.	DMA block diagram	163
Figure 21.	DMA1 request mapping	170
Figure 22.	DMA2 request mapping	172
Figure 23.	EXTI extended interrupt/event block diagram	187
Figure 24.	Extended interrupt/event GPIO mapping	189
Figure 25.	Single ADC block diagram	197
Figure 26.	Timing diagram	200
Figure 27.	Analog watchdog guarded area	200
Figure 28.	Injected conversion latency	202
Figure 29.	Calibration timing diagram	203
Figure 30.	Right alignment of data	204
Figure 31.	Left alignment of data	204
Figure 32.	Temperature sensor and VREFINT channel block diagram	206
Figure 33.	SDADC clock block diagram	225
Figure 34.	Single SDADC block diagram	226
Figure 35.	Switch configuration in single-ended mode	229
Figure 36.	Switch configuration in differential mode	230
Figure 37.	Switch configuration in mixed mode (example 1)	231
Figure 38.	Switch configuration in mixed mode (example 2)	232
Figure 39.	Equivalent input circuit for input channel	237
Figure 40.	Equivalent input circuit for VREFSD input	237
Figure 41.	DAC1 block diagram	262
Figure 42.	DAC2 block diagram	263
Figure 43.	Data registers in single DAC channel mode	264
Figure 44.	Timing diagram for conversion with trigger disabled $TEN = 0$	265
Figure 45.	Data registers in single DAC channel mode	268
Figure 46.	Data registers in dual DAC channel mode	268
Figure 47.	DAC LFSR register calculation algorithm	273
Figure 48.	DAC conversion (SW trigger enabled) with LFSR wave generation	274

Figure 49.	DAC triangle wave generation . . . . .	274
Figure 50.	DAC conversion (SW trigger enabled) with triangle wave generation . . . . .	275
Figure 51.	Comparator 1 and 2 block diagrams. . . . .	288
Figure 52.	Comparator hysteresis . . . . .	289
Figure 53.	General-purpose timer block diagram . . . . .	296
Figure 54.	Counter timing diagram with prescaler division change from 1 to 2 . . . . .	297
Figure 55.	Counter timing diagram with prescaler division change from 1 to 4 . . . . .	298
Figure 56.	Counter timing diagram, internal clock divided by 1 . . . . .	299
Figure 57.	Counter timing diagram, internal clock divided by 2 . . . . .	299
Figure 58.	Counter timing diagram, internal clock divided by 4 . . . . .	300
Figure 59.	Counter timing diagram, internal clock divided by N . . . . .	300
Figure 60.	Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded). . . . .	301
Figure 61.	Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded). . . . .	301
Figure 62.	Counter timing diagram, internal clock divided by 1 . . . . .	302
Figure 63.	Counter timing diagram, internal clock divided by 2 . . . . .	303
Figure 64.	Counter timing diagram, internal clock divided by 4 . . . . .	303
Figure 65.	Counter timing diagram, internal clock divided by N . . . . .	304
Figure 66.	Counter timing diagram, Update event when repetition counter is not used . . . . .	304
Figure 67.	Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . . .	306
Figure 68.	Counter timing diagram, internal clock divided by 2 . . . . .	306
Figure 69.	Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . .	307
Figure 70.	Counter timing diagram, internal clock divided by N . . . . .	307
Figure 71.	Counter timing diagram, Update event with ARPE=1 (counter underflow). . . . .	308
Figure 72.	Counter timing diagram, Update event with ARPE=1 (counter overflow). . . . .	308
Figure 73.	Control circuit in normal mode, internal clock divided by 1 . . . . .	309
Figure 74.	TI2 external clock connection example. . . . .	310
Figure 75.	Control circuit in external clock mode 1 . . . . .	311
Figure 76.	External trigger input block . . . . .	311
Figure 77.	Control circuit in external clock mode 2 . . . . .	312
Figure 78.	Capture/compare channel (example: channel 1 input stage). . . . .	313
Figure 79.	Capture/compare channel 1 main circuit . . . . .	313
Figure 80.	Output stage of capture/compare channel (channel 1). . . . .	314
Figure 81.	PWM input mode timing . . . . .	316
Figure 82.	Output compare mode, toggle on OC1. . . . .	318
Figure 83.	Edge-aligned PWM waveforms (ARR=8) . . . . .	319
Figure 84.	Center-aligned PWM waveforms (ARR=8). . . . .	320
Figure 85.	Example of one-pulse mode. . . . .	321
Figure 86.	Clearing TIMx_OCxREF . . . . .	323
Figure 87.	Example of counter operation in encoder interface mode . . . . .	325
Figure 88.	Example of encoder interface mode with TI1FP1 polarity inverted . . . . .	325
Figure 89.	Control circuit in reset mode . . . . .	326
Figure 90.	Control circuit in gated mode . . . . .	327
Figure 91.	Control circuit in trigger mode. . . . .	328
Figure 92.	Control circuit in external clock mode 2 + trigger mode . . . . .	329
Figure 93.	Master/Slave timer example . . . . .	329
Figure 94.	Gating timer y with OC1REF of timer x. . . . .	330
Figure 95.	Gating timer y with Enable of timer x . . . . .	331
Figure 96.	Triggering timer y with update of timer x. . . . .	332
Figure 97.	Triggering timer y with Enable of timer x . . . . .	332
Figure 98.	Triggering timer x and y with timer x TI1 input . . . . .	334
Figure 99.	General-purpose timer block diagram (TIM12) . . . . .	359

Figure 100.	General-purpose timer block diagram (TIM13/14)	360
Figure 101.	Counter timing diagram with prescaler division change from 1 to 2	362
Figure 102.	Counter timing diagram with prescaler division change from 1 to 4	363
Figure 103.	Counter timing diagram, internal clock divided by 1	364
Figure 104.	Counter timing diagram, internal clock divided by 2	365
Figure 105.	Counter timing diagram, internal clock divided by 4	365
Figure 106.	Counter timing diagram, internal clock divided by N	366
Figure 107.	Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)	366
Figure 108.	Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	367
Figure 109.	Control circuit in normal mode, internal clock divided by 1	368
Figure 110.	TI2 external clock connection example	368
Figure 111.	Control circuit in external clock mode 1	369
Figure 112.	Capture/compare channel (example: channel 1 input stage)	370
Figure 113.	Capture/compare channel 1 main circuit	370
Figure 114.	Output stage of capture/compare channel (channel 1)	371
Figure 115.	PWM input mode timing	373
Figure 116.	Output compare mode, toggle on OC1	374
Figure 117.	Edge-aligned PWM waveforms (ARR=8)	375
Figure 118.	Example of one pulse mode	376
Figure 119.	Control circuit in reset mode	378
Figure 120.	Control circuit in gated mode	379
Figure 121.	Control circuit in trigger mode	379
Figure 122.	TIM15 block diagram	408
Figure 123.	TIM16 and TIM17 block diagram	409
Figure 124.	Counter timing diagram with prescaler division change from 1 to 2	411
Figure 125.	Counter timing diagram with prescaler division change from 1 to 4	411
Figure 126.	Counter timing diagram, internal clock divided by 1	413
Figure 127.	Counter timing diagram, internal clock divided by 2	413
Figure 128.	Counter timing diagram, internal clock divided by 4	414
Figure 129.	Counter timing diagram, internal clock divided by N	414
Figure 130.	Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)	415
Figure 131.	Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	415
Figure 132.	Update rate examples depending on mode and TIMx_RCR register settings	417
Figure 133.	Control circuit in normal mode, internal clock divided by 1	418
Figure 134.	TI2 external clock connection example	418
Figure 135.	Control circuit in external clock mode 1	419
Figure 136.	Capture/compare channel (example: channel 1 input stage)	420
Figure 137.	Capture/compare channel 1 main circuit	420
Figure 138.	Output stage of capture/compare channel (channel 1)	421
Figure 139.	Output stage of capture/compare channel (channel 2 for TIM15)	421
Figure 140.	PWM input mode timing	423
Figure 141.	Output compare mode, toggle on OC1	425
Figure 142.	Edge-aligned PWM waveforms (ARR=8)	426
Figure 143.	Complementary output with dead-time insertion	427
Figure 144.	Dead-time waveforms with delay greater than the negative pulse	428
Figure 145.	Dead-time waveforms with delay greater than the positive pulse	428
Figure 146.	Output behavior in response to a break	431
Figure 147.	Example of one pulse mode	432

Figure 148. Control circuit in reset mode . . . . .	434
Figure 149. Control circuit in gated mode . . . . .	435
Figure 150. Control circuit in trigger mode . . . . .	436
Figure 151. IR internal hardware connections with TIM16 and TIM17. . . . .	478
Figure 152. Basic timer block diagram . . . . .	479
Figure 153. Counter timing diagram with prescaler division change from 1 to 2 . . . . .	481
Figure 154. Counter timing diagram with prescaler division change from 1 to 4 . . . . .	481
Figure 155. Counter timing diagram, internal clock divided by 1 . . . . .	482
Figure 156. Counter timing diagram, internal clock divided by 2 . . . . .	483
Figure 157. Counter timing diagram, internal clock divided by 4 . . . . .	483
Figure 158. Counter timing diagram, internal clock divided by N. . . . .	484
Figure 159. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). . . . .	484
Figure 160. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded). . . . .	485
Figure 161. Control circuit in normal mode, internal clock divided by 1 . . . . .	486
Figure 162. Independent watchdog block diagram . . . . .	492
Figure 163. Watchdog block diagram . . . . .	501
Figure 164. Window watchdog timing diagram . . . . .	502
Figure 165. RTC block diagram . . . . .	508
Figure 166. I2C block diagram . . . . .	553
Figure 167. I2C bus protocol . . . . .	555
Figure 168. Setup and hold timings . . . . .	557
Figure 169. I2C initialization flowchart . . . . .	560
Figure 170. Data reception . . . . .	561
Figure 171. Data transmission . . . . .	562
Figure 172. Slave initialization flowchart . . . . .	565
Figure 173. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0 . . . . .	567
Figure 174. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1 . . . . .	568
Figure 175. Transfer bus diagrams for I2C slave transmitter. . . . .	569
Figure 176. Transfer sequence flowchart for slave receiver with NOSTRETCH=0 . . . . .	570
Figure 177. Transfer sequence flowchart for slave receiver with NOSTRETCH=1 . . . . .	571
Figure 178. Transfer bus diagrams for I2C slave receiver. . . . .	571
Figure 179. Master clock generation . . . . .	573
Figure 180. Master initialization flowchart . . . . .	575
Figure 181. 10-bit address read access with HEAD10R=0 . . . . .	575
Figure 182. 10-bit address read access with HEAD10R=1 . . . . .	576
Figure 183. Transfer sequence flowchart for I2C master transmitter for N<=255 bytes . . . . .	577
Figure 184. Transfer sequence flowchart for I2C master transmitter for N>255 bytes . . . . .	578
Figure 185. Transfer bus diagrams for I2C master transmitter . . . . .	579
Figure 186. Transfer sequence flowchart for I2C master receiver for N<=255 bytes . . . . .	581
Figure 187. Transfer sequence flowchart for I2C master receiver for N >255 bytes . . . . .	582
Figure 188. Transfer bus diagrams for I2C master receiver . . . . .	583
Figure 189. Timeout intervals for $t_{LOW:SEXT}$ , $t_{LOW:MEXT}$ . . . . .	588
Figure 190. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC. . . . .	592
Figure 191. Transfer bus diagrams for SMBus slave transmitter (SBC=1) . . . . .	592
Figure 192. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC . . . . .	594
Figure 193. Bus transfer diagrams for SMBus slave receiver (SBC=1) . . . . .	595
Figure 194. Bus transfer diagrams for SMBus master transmitter. . . . .	596
Figure 195. Bus transfer diagrams for SMBus master receiver . . . . .	598
Figure 196. I2C interrupt mapping diagram . . . . .	603
Figure 197. USART block diagram . . . . .	625

Figure 198. Word length programming . . . . .	626
Figure 199. Configurable stop bits . . . . .	627
Figure 200. TC/TXE behavior when transmitting . . . . .	629
Figure 201. Start bit detection when oversampling by 16 or 8 . . . . .	630
Figure 202. Data sampling when oversampling by 16 . . . . .	633
Figure 203. Data sampling when oversampling by 8 . . . . .	633
Figure 204. Mute mode using Idle line detection . . . . .	640
Figure 205. Mute mode using address mark detection . . . . .	641
Figure 206. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . . .	644
Figure 207. Break detection in LIN mode vs. Framing error detection . . . . .	645
Figure 208. USART example of synchronous transmission . . . . .	646
Figure 209. USART data clock timing diagram (M=0) . . . . .	646
Figure 210. USART data clock timing diagram (M=1) . . . . .	647
Figure 211. RX data setup/hold time . . . . .	647
Figure 212. ISO 7816-3 asynchronous protocol . . . . .	648
Figure 213. Parity error detection using the 1.5 stop bits . . . . .	650
Figure 214. IrDA SIR ENDEC- block diagram . . . . .	654
Figure 215. IrDA data modulation (3/16) -Normal Mode . . . . .	654
Figure 216. Transmission using DMA . . . . .	656
Figure 217. Reception using DMA . . . . .	657
Figure 218. Hardware flow control between 2 USARTs . . . . .	657
Figure 219. RTS flow control . . . . .	658
Figure 220. CTS flow control . . . . .	658
Figure 221. USART interrupt mapping diagram . . . . .	661
Figure 222. SPI block diagram . . . . .	688
Figure 223. Full-duplex single master/ single slave application . . . . .	689
Figure 224. Half-duplex single master/ single slave application . . . . .	690
Figure 225. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) . . . . .	691
Figure 226. Master and three independent slaves . . . . .	692
Figure 227. Hardware/software slave select management . . . . .	693
Figure 228. Data clock timing diagram . . . . .	695
Figure 229. Data alignment when data length is not equal to 8-bit or 16-bit . . . . .	696
Figure 230. Packing data in FIFO for transmission and reception . . . . .	700
Figure 231. Master full duplex communication . . . . .	703
Figure 232. Slave full duplex communication . . . . .	704
Figure 233. Master full duplex communication with CRC . . . . .	705
Figure 234. Master full duplex communication in packed mode . . . . .	706
Figure 235. NSSP pulse generation in Motorola SPI master mode . . . . .	709
Figure 236. TI mode transfer . . . . .	710
Figure 237. I <sup>2</sup> S block diagram . . . . .	713
Figure 238. I <sup>2</sup> S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0) . . . . .	715
Figure 239. I <sup>2</sup> S Philips standard waveforms (24-bit frame with CPOL = 0) . . . . .	715
Figure 240. Transmitting 0x8EAA33 . . . . .	716
Figure 241. Receiving 0x8EAA33 . . . . .	716
Figure 242. I <sup>2</sup> S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0) . . . . .	716
Figure 243. Example of 16-bit data frame extended to 32-bit channel frame . . . . .	716
Figure 244. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0 . . . . .	717
Figure 245. MSB justified 24-bit frame length with CPOL = 0 . . . . .	717
Figure 246. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0 . . . . .	717
Figure 247. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0 . . . . .	718

Figure 248. LSB justified 24-bit frame length with CPOL = 0 . . . . .	718
Figure 249. Operations required to transmit 0x3478AE . . . . .	718
Figure 250. Operations required to receive 0x3478AE . . . . .	719
Figure 251. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0 . . . . .	719
Figure 252. Example of 16-bit data frame extended to 32-bit channel frame . . . . .	719
Figure 253. PCM standard waveforms (16-bit) . . . . .	720
Figure 254. PCM standard waveforms (16-bit extended to 32-bit packet frame). . . . .	720
Figure 255. Audio sampling frequency definition . . . . .	721
Figure 256. I <sup>2</sup> S clock generator architecture . . . . .	721
Figure 257. TSC block diagram . . . . .	744
Figure 258. Surface charge transfer analog I/O group structure . . . . .	745
Figure 259. Sampling capacitor voltage variation . . . . .	746
Figure 260. Charge transfer acquisition sequence . . . . .	747
Figure 261. Spread spectrum variation principle . . . . .	748
Figure 262. CAN network topology . . . . .	763
Figure 263. bxCAN operating modes. . . . .	766
Figure 264. bxCAN in silent mode . . . . .	767
Figure 265. bxCAN in loop back mode . . . . .	767
Figure 266. bxCAN in combined mode . . . . .	768
Figure 267. Transmit mailbox states . . . . .	769
Figure 268. Receive FIFO states . . . . .	770
Figure 269. Filter bank scale configuration - register organization . . . . .	773
Figure 270. Example of filter numbering . . . . .	774
Figure 271. Filtering mechanism - example . . . . .	775
Figure 272. CAN error state diagram. . . . .	776
Figure 273. Bit timing . . . . .	778
Figure 274. CAN frames . . . . .	779
Figure 275. Event flags and interrupt generation. . . . .	780
Figure 276. Can mailbox registers . . . . .	792
Figure 277. USB peripheral block diagram . . . . .	808
Figure 278. Packet buffer areas with examples of buffer description table locations . . . . .	812
Figure 279. Block diagram . . . . .	838
Figure 280. Message structure . . . . .	839
Figure 281. Blocks . . . . .	839
Figure 282. Bit timings . . . . .	840
Figure 283. Signal free time . . . . .	840
Figure 284. Arbitration phase. . . . .	841
Figure 285. SFT of three nominal bit periods. . . . .	841
Figure 286. Error bit timing . . . . .	842
Figure 287. Error handling . . . . .	844
Figure 288. TXERR detection . . . . .	845
Figure 289. Block diagram of STM32F37xx MCU and Cortex <sup>®</sup> -M4 with FPU-level debug support . . . . .	856
Figure 290. SWJ debug port . . . . .	858
Figure 291. JTAG TAP connections . . . . .	862
Figure 292. TPIU block diagram . . . . .	881

# 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.
Reserved (Res.)	Reserved bit, must be kept at reset value.

## 1.2 Glossary

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

- The Cortex<sup>®</sup>-M4 with FPU core integrates one debug port: **SWD debug port (SWD-DP)** provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol. Please refer to the Cortex<sup>®</sup>-M4 with FPU technical reference manual.
- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **IAP (in-application programming)**: IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running.
- **ICP (in-circuit programming)**: ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the bootloader while the device is mounted on the user application board.
- **Option bytes**: product configuration bits stored in the Flash memory.
- **OBL**: option byte loader.
- **AHB**: advanced high-performance bus.

## 1.3 Peripheral availability

For peripheral availability and number across all sales types, please refer to the particular device datasheet.

## 2 System architecture and memory overview

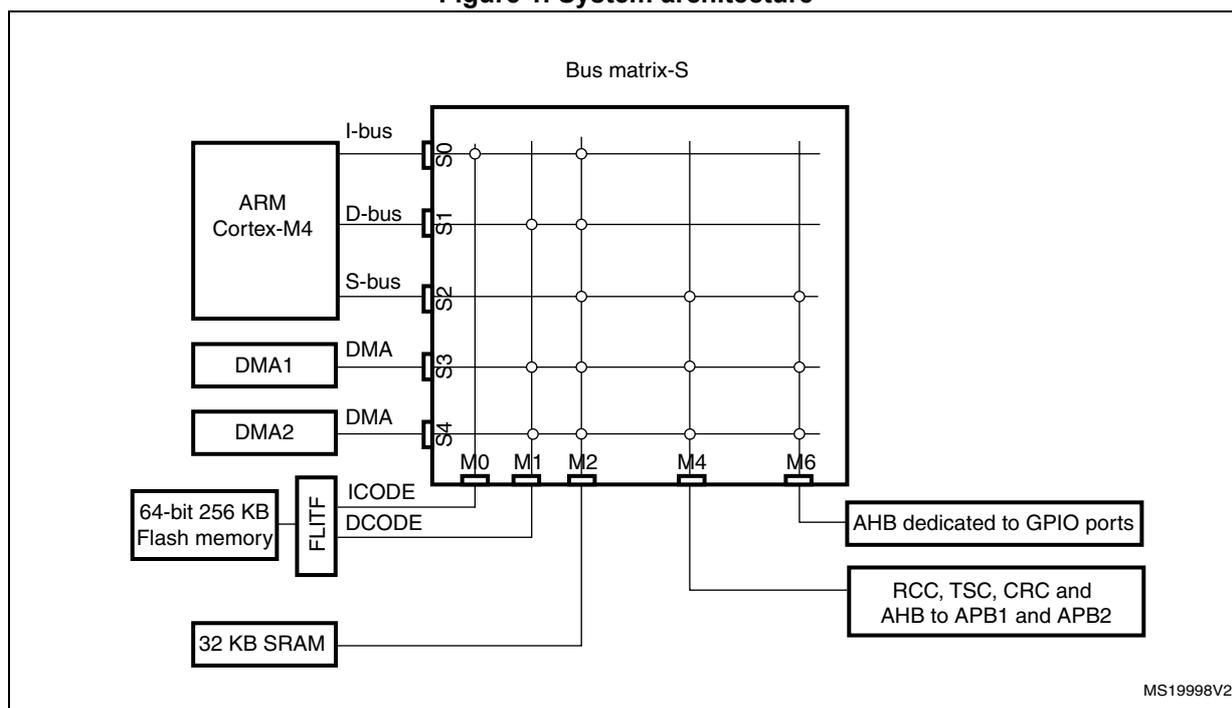
### 2.1 System architecture

The main system consists of:

- Five masters:
  - Cortex<sup>®</sup>-M4 with FPU core I-bus
  - Cortex<sup>®</sup>-M4 with FPU core D-bus
  - Cortex<sup>®</sup>-M4 with FPU core S-bus
  - GP-DMA1 and GP-DMA2 (general-purpose DMAs)
- Five slaves:
  - Internal SRAM
  - Internal Flash memory (ICODE and DCODE)
  - AHB to APBx (APB1 or APB2), which connect all the APB peripherals
  - AHB dedicated to GPIO ports

These are interconnected using a multilayer AHB bus architecture as shown in [Figure 1](#):

**Figure 1. System architecture**



#### 2.1.1 S0: I-bus

This bus connects the Instruction bus of the Cortex<sup>®</sup>-M4 with FPU core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory area (Flash and SRAM) containing the code.

### 2.1.2 S1: D-bus

This bus connects the Data bus of the Cortex<sup>®</sup>-M4 with FPU core to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory area (Flash and SRAM) containing the code or data.

### 2.1.3 S2: S-bus

This bus connects the system bus of the Cortex<sup>®</sup>-M4 with FPU core to the BusMatrix. This bus is used to access data located in peripheral or SRAM area. Instructions can also be fetched on this bus even if it less efficient than the ICode bus.

The targets of this bus are the 32-Kbyte SRAM, the AHB2APB bridges, the AHB I/O port.

### 2.1.4 S3, S4: DMA-bus

This bus connects the AHB master interface of the DMAs to the BusMatrix which manages the access of different Masters to Flash memory, SRAM and AHB peripherals.

### 2.1.5 BusMatrix-S (5M5S)

The BusMatrix manages the access arbitration between Masters (core system bus, GP-DMAs). The arbitration scheme uses a Round Robin algorithm. The BusMatrix is composed of 5 slaves (FLASH ITF, SRAM, AHB2APB bridges and AHB I/O ports) and 5 masters (CPU System, DCODE and ICODE buses, DMA1 and DMA2 bus).

The following subsections describe all the peripherals connected to the AHB subsystem.

#### AHB/APB bridges

The two AHB/APB bridges provide full synchronous connections between the AHB and the 2 APB buses. APB1 is limited to 36 MHz, APB2 operates at full speed(72 MHz).

Refer to [Section 2.2.2: Memory map and register boundary addresses on page 39](#) for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM and 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

### 2.2.1 Introduction

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

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

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

All the memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved". For the detailed mapping of available memory and register areas, please refer to the [Memory map and register boundary addresses](#) chapter and peripheral chapters.

### 2.2.2 Memory map and register boundary addresses

See the datasheet corresponding to your device for a comprehensive diagram of the memory map.

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

**Table 1. STM32F37xx peripheral register boundary addresses**

Bus	Boundary address	Size	Peripheral	Peripheral register map
	0xE000 0000 - 0xE010 0000	1MB	Cortex <sup>®</sup> -M4 with FPU internal peripherals	
	0x4800 1800 - 0x5FFF FFFF	~384 MB	Reserved	
AHB2	0x4800 1400 - 0x4800 17FF	1KB	GPIOF	<a href="#">Section 8.4.11 on page 152</a>
	0x4800 1000 - 0x4800 13FF	1KB	GPIOE	<a href="#">Section 8.4.11 on page 152</a>
	0x4800 0C00 - 0x4800 0FFF	1KB	GIOD	<a href="#">Section 8.4.11 on page 152</a>
	0x4800 0800 - 0x4800 0BFF	1KB	GPIOC	<a href="#">Section 8.4.11 on page 152</a>
	0x4800 0400 - 0x4800 07FF	1KB	GPIOB	<a href="#">Section 8.4.11 on page 152</a>
	0x4800 0000 - 0x4800 03FF	1KB	GPIOA	<a href="#">Section 8.4.11 on page 152</a>
	0x4002 4400 - 0x47FF FFFF	~128 MB	Reserved	

Table 1. STM32F37xx peripheral register boundary addresses (continued)

Bus	Boundary address	Size	Peripheral	Peripheral register map
AHB	0x4002 4000 - 0x4002 43FF	1 KB	TSC	<a href="#">Section 27.6.11 on page 760</a>
	0x4002 3400 - 0x4002 3FFF	3 KB	Reserved	
	0x4002 3000 - 0x4002 33FF	1 KB	CRC	<a href="#">Section 5.4.6 on page 74</a>
	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	
	0x4002 2000 - 0x4002 23FF	1 KB	FLASH memory interface	<a href="#">Section 3.6 on page 64</a>
	0x4002 1400 - 0x4002 1FFF	3 KB	Reserved	
	0x4002 1000 - 0x4002 13FF	1 KB	RCC	<a href="#">Section 7.4.14 on page 135</a>
	0x4002 0800 - 0x4002 0FFF	2 KB	Reserved	
	0x4002 0400 - 0x4002 07FF	1 KB	DMA2	<a href="#">Section 10.5.7 on page 180</a>
	0x4002 0000 - 0x4002 03FF	1 KB	DMA1	<a href="#">Section 10.5.7 on page 180</a>
	0x4001 6C00 - 0x4001 FFFF	37 KB	Reserved	
APB2	0x4001 6800 - 0x4001 6BFF	1 KB	SDADC3	<a href="#">Section 13.6.18 on page 259</a>
	0x4001 6400 - 0x4001 67FF	1 KB	SDADC2	<a href="#">Section 13.6.18 on page 259</a>
	0x4001 6000 - 0x4001 63FF	1 KB	SDADC1	<a href="#">Section 13.6.18 on page 259</a>
	0x4001 5C00 - 0x4001 5FFF	1 KB	TIM19	<a href="#">Section 16.4.20 on page 355</a>
	0x4001 5800 - 0x4001 5BFF	1 KB	DBGMCU	<a href="#">Section 31.18: DBG register map on page 886</a>
	0x4001 4C00 - 0x4001 57FF	4 KB	Reserved	
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	<a href="#">Section 18.6.16 on page 476</a>
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	<a href="#">Section 18.6.16 on page 476</a>
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	<a href="#">Section 18.5.18 on page 457</a>
	0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved	
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	<a href="#">Section 25.7.12 on page 685</a>
	0x4001 3400 - 0x4001 37FF	1 KB	Reserved	
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1/I2S1	<a href="#">Section 26.9.10 on page 742</a>
	0x4001 2800 - 0x4001 2FFF	1 KB	Reserved	
	0x4001 2400 - 0x4001 27FF	1 KB	ADC	<a href="#">Section 12.12.15 on page 221</a>
	0x4001 0800 - 0x4001 23FF	7 KB	Reserved	
	0x4001 0400 - 0x4001 07FF	1 KB	EXTI	<a href="#">Section 11.3.7 on page 195</a>
	0x4001 0000 - 0x4001 03FF	1 KB	SYSCFG + COMP	<a href="#">Section 9.1.7 on page 161</a> and <a href="#">Section 15.6.2: COMP register map</a>
	0x4000 A000 - 0x4000 FFFF	32 KB	Reserved	

**Table 1. STM32F37xx peripheral register boundary addresses (continued)**

Bus	Boundary address	Size	Peripheral	Peripheral register map
APB1	0x4000 9C00 - 0x4000 9FFF	1 KB	TIM18	<a href="#">Section 20.4.9 on page 491</a>
	0x4000 9800 - 0x4000 9BFF	1 KB	DAC2	<a href="#">Section 14.8.15 on page 285</a>
	0x4000 7C00 - 0x4000 97FF	7 KB	Reserved	
	0x4000 7800 - 0x4000 7BFF	1 KB	CEC	<a href="#">Section 30.7.7 on page 855</a>
	0x4000 7400 - 0x4000 77FF	1 KB	DAC1	<a href="#">Section 14.8.15 on page 285</a>
	0x4000 7000 - 0x4000 73FF	1 KB	PWR	<a href="#">Section 6.4.3 on page 93</a>
	0x4000 6800 - 0x4000 6FFF	2 KB	Reserved	
	0x4000 6400 - 0x4000 67FF	1 KB	CAN	<a href="#">Section 28.9.5 on page 803</a>
	0x4000 6200 - 0x4000 63FF	1 KB	Reserved	
	0x4000 6000 - 0x4000 61FF	0.5 KB	USB packet (stored in SRAM)	<a href="#">Section 29.6.4 on page 835</a>
	0x4000 5C00 - 0x4000 5FFF	1 KB	USB FS	<a href="#">Section 29.6.4 on page 835</a>
	0x4000 5800 - 0x4000 5BFF	1 KB	I2C2	<a href="#">Section 24.7 on page 603</a>
	0x4000 5400 - 0x4000 57FF	1 KB	I2C1	<a href="#">Section 24.7 on page 603</a>
	0x4000 4C00 - 0x4000 53FF	2 KB	Reserved	
	0x4000 4800 - 0x4000 4BFF	1 KB	USART3	<a href="#">Section 25.7.12 on page 685</a>
	0x4000 4400 - 0x4000 47FF	1 KB	USART2	<a href="#">Section 25.7.12 on page 685</a>
	0x4000 4000 - 0x4000 43FF	1 KB	Reserved	
	0x4000 3C00 - 0x4000 3FFF	1 KB	SPI3/I2S3	<a href="#">Section 26.9.10 on page 742</a>
	0x4000 3800 - 0x4000 3BFF	1 KB	SPI2/I2S2	<a href="#">Section 26.9.10 on page 742</a>
	0x4000 3400 - 0x4000 37FF	1 KB	Reserved	
0x4000 3000 - 0x4000 33FF	1 KB	IWDG	<a href="#">Section 21.4.6 on page 499</a>	
0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG	<a href="#">Section 22.4.4 on page 505</a>	
0x4000 2800 - 0x4000 2BFF	1 KB	RTC	<a href="#">Section 23.6.20 on page 549</a>	
APB1	0x4000 2400 - 0x4000 27FF	1 KB	Reserved	
	0x4000 2000 - 0x4000 23FF	1 KB	TIM14	<a href="#">Section 17.6.12 on page 404</a>
	0x4000 1C00 - 0x4000 1FFF	1 KB	TIM13	<a href="#">Section 17.6.12 on page 404</a>
	0x4000 1800 - 0x4000 1BFF	1 KB	TIM12	<a href="#">Section 17.5.13 on page 393</a>
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7	<a href="#">Section 20.4.9 on page 491</a>
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6	<a href="#">Section 20.4.9 on page 491</a>
	0x4000 0C00 - 0x4000 0FFF	1 KB	TIM5	<a href="#">Section 16.4.20 on page 355</a>
	0x4000 0800 - 0x4000 0BFF	1 KB	TIM4	<a href="#">Section 16.4.20 on page 355</a>
	0x4000 0400 - 0x4000 07FF	1 KB	TIM3	<a href="#">Section 16.4.20 on page 355</a>
	0x4000 0000 - 0x4000 03FF	1 KB	TIM2	<a href="#">Section 16.4.20 on page 355</a>
	0x2000 8000 - 3FFF FFFF	~512 MB	Reserved	
	0x2000 0000 - 0x2000 7FFF	32 KB	SRAM	

**Table 1. STM32F37xx peripheral register boundary addresses (continued)**

Bus	Boundary address	Size	Peripheral	Peripheral register map
	0x1FFF FC00 - 0x1FFF FFFF	1 KB	Reserved	
	0x1FFF F800 - 0x1FFF FBFF	1 KB	Option bytes	
	0x1FFF D800 - 0x1FFF F7FF	8 KB	System memory	
	0x0801 0000 - 0x1FFF EBFF	~384 MB	Reserved	
	0x0800 0000 - 0x0803 FFFF	256 KB	Main Flash memory	
	0x0001 0000 - 0x07FF FFFF	128 MB	Reserved	
	0x0000 000 - 0x0000 FFFF	64 KB	Main Flash memory, system memory or SRAM depending on BOOT configuration	

## 2.3 Embedded SRAM

The STM32F37xx features 32 Kbytes of static SRAM. It can be accessed as bytes, half-words (16 bits) or full words (32 bits). This memory can be addressed at maximum system clock frequency without wait state and thus by both CPU and DMA.

### 2.3.1 Parity check

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

The parity bits are computed and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated. The same error can also be linked to the BRK\_IN Break input of TIMER1, with the SRAM\_PARITY\_LOCK control bit in the [SYSCFG configuration register 2 \(SYSCFG\\_CFGR2\)](#). The SRAM Parity Error flag (SRAM\_PEF) is available in the [SYSCFG configuration register 2 \(SYSCFG\\_CFGR2\)](#).

## 2.4 Bit banding

The Cortex<sup>®</sup>-M4 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 STM32F37xx both peripheral registers and SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed. The operations are only available for Cortex<sup>®</sup>-M4 accesses, not from other bus masters (e.g. DMA).

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:

$$bit\_word\_addr = bit\_band\_base + (byte\_offset \times 32) + (bit\_number \times 4)$$

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 0x20000300 in the alias region:

$$0x22006008 = 0x22000000 + (0x300 \times 32) + (2 \times 4).$$

Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x20000300.

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM address 0x20000300 (0x01: bit set; 0x00: bit reset).

For more information on Bit-Banding, please refer to the *Cortex<sup>®</sup>-M4 Technical Reference Manual*.

## 2.5 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block. It contains the application program and user data if necessary.
- The information block. It is composed of two parts:
  - Option bytes for hardware and memory protection user configuration.
  - System memory which contains the proprietary bootloader code. Please, refer to [Section 3: Embedded Flash memory](#) for more details.

Flash memory instructions and data access are performed through the AHB bus. The prefetch block is used for instruction fetches through the ICode bus. Arbitration is performed in the Flash memory interface, and priority is given to data access on the DCode bus. It also implements the logic necessary to carry out the Flash memory operations (Program/Erase) controlled through the Flash registers.

## 2.6 Boot configuration

In the STM32F37xx, three different boot modes can be selected through the BOOT0 pin and nBOOT1 bit in the User option byte, as shown in the following table:

**Table 2. Boot modes**

Boot mode selection		Boot mode	Aliasing
BOOT1 (inverted nBOOT1)	BOOT0		
x	0	Main Flash memory	Main flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM (on the DCode bus) is selected as boot space

The values on both BOOT0 pin and nBOOT1 option bit are latched on the 4th rising edge of SYSCLK after a reset.

It is up to the user to set the nBOOT1 and BOOT0 to select the required boot mode. The BOOT0 pin and nBOOT1 option bit are also resampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory at 0x0000 0004. Depending on the selected boot mode, main Flash memory, system memory or SRAM is accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x0800

0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x0800 0000.

- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF D800).
- Boot from the embedded SRAM: the SRAM is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x2000 0000).

### **Embedded bootloader**

The embedded bootloader is located in the System memory, programmed by ST during production. It is used to reprogram the Flash memory through USART1 or USART2 or USB (DFU: device firmware upgrade).

### 3 Embedded Flash memory

#### 3.1 Flash main features

- Up to 256 Kbytes of Flash memory
- Memory organization:
  - Main memory block:  
32 Kbits × 64 bits
  - Information block:  
1280 × 64 bits

Flash memory interface (FLITF) features:

- Read interface with prefetch buffer (2 × 64-bit words)
- Option byte loader
- Flash program/Erase operation
- Read/Write protection
- Low-power mode

#### 3.2 Flash memory functional description

##### 3.2.1 Flash memory organization

The Flash memory is organized as 64-bit wide memory cells that can be used for storing both code and data constants.

The memory organization is based on a main memory block containing 128 pages of 2 Kbyte and an information block as shown in [Table 3](#).

**Table 3. Flash module organization**

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2 K	Page 0
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 1800 - 0x0800 1FFF	2 K	Page 3
	.	.	.
	.	.	.
	.	.	.
	.	.	.
	.	.	.
	.	.	.
	.	..	.
	.	.	.
	.	.	.
	0x0803 F800 - 0x0803 FFFF	2 K	Page 127

Table 3. Flash module organization (continued)

Flash area	Flash memory addresses	Size (bytes)	Name
Information block	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory
	0x1FFF F800 - 0x1FFF F80F	16	Option bytes
Flash memory interface registers	0x4002 2000 - 0x4002 2003	4	FLASH_ACR
	0x4002 2004 - 0x4002 2007	4	FLASH_KEYR
	0x4002 2008 - 0x4002 200B	4	FLASH_OPTKEYR
	0x4002 200C - 0x4002 200F	4	FLASH_SR
	0x4002 2010 - 0x4002 2013	4	FLASH_CR
	0x4002 2014 - 0x4002 2017	4	FLASH_AR
	0x4002 2018 - 0x4002 201B	4	Reserved
	0x4002 201C - 0x4002 201F	4	FLASH_OBR
	0x4002 2020 - 0x4002 2023	4	FLASH_WRPR

The information block is divided into two parts:

- System memory is used to boot the device in System memory boot mode. The area is reserved for use by STMicroelectronics and contains the bootloader which is used to reprogram the Flash memory through one of the following interfaces: USART1, USART2 or USB (DFU). It is programmed by ST when the device is manufactured, and protected against spurious write/erase operations. For further details, please refer to the AN2606.
- Option bytes

### 3.2.2 Read operations

The embedded Flash module can be addressed directly, as a common memory space. Any data read operation accesses the content of the Flash module through dedicated read senses and provides the requested data.

The read interface consists of a read controller on one side to access the Flash memory and an AHB interface on the other side to interface with the CPU. The main task of the read interface is to generate the control signals to read from the Flash memory and to prefetch the blocks required by the CPU. The prefetch block is only used for instruction fetches over the ICode bus. The Literal pool is accessed over the DCode bus. Since these two buses have the same Flash memory as target, DCode bus accesses have priority over prefetch accesses.

Read accesses can be performed with the following options managed through the Flash access control register (FLASH\_ACR):

- Instruction fetch: Prefetch buffer enabled for a faster CPU execution.
- Latency: number of wait states for a correct read operation (from 0 to 2)

### Instruction fetch

The Cortex<sup>®</sup>-M4 with FPU fetches the instruction over the ICode bus and the literal pool (constant/data) over the DCode bus. The prefetch block aims at increasing the efficiency of ICode bus accesses.

### Prefetch buffer

The prefetch controller decides to access the Flash memory depending on the available space in the prefetch buffer. The controller initiates a read request when there is at least one block free in the prefetch buffer.

After reset, the state of the prefetch buffer is on. The prefetch buffer is usually switched on/off during the initialization routine, while the microcontroller is running on the internal 8 MHz RC (HSI) oscillator.

### Prefetch controller

The prefetch controller decides to access the Flash memory depending on the available space in the prefetch buffer. The Controller initiates a read request when there is at least one block free in the prefetch buffer.

After reset, the state of the prefetch buffer is on. The prefetch buffer should be switched on/off only when SYSCLK is lower than 24 MHz and no prescaler is applied on the AHB clock (SYSCLK must be equal to HCLK). The prefetch buffer is usually switched on/off during the initialization routine, while the microcontroller is running on the internal 8 MHz RC (HSI) oscillator.

*Note:* The prefetch buffer must be kept on (FLASH\_ACR[4]='1') when using a prescaler different from 1 on the AHB clock.

### Access latency

In order to maintain the control signals to read the Flash memory, the ratio of the prefetch controller clock period to the access time of the Flash memory has to be programmed in the Flash access control register with the LATENCY[2:0] bits. This value gives the number of cycles needed to maintain the control signals of the Flash memory and correctly read the required data. After reset, the value is zero and only one cycle without additional wait states is required to access the Flash memory.

### DCode interface

The DCode interface consists of a simple AHB interface on the CPU side and a request generator to the Arbiter of the Flash access controller. The DCode accesses have priority over prefetch accesses. This interface uses the Access Time Tuner block of the prefetch buffer.

### Flash Access controller

Mainly, this block is a simple arbiter between the read requests of the prefetch/ICode and DCode interfaces.

DCode interface requests have priority over other requests.

### 3.2.3 Flash program and erase operations

The STM32F37xx embedded Flash memory can be programmed using in-circuit programming or in-application programming.

The **in-circuit programming (ICP)** method is used to update the entire contents of the Flash memory, using the JTAG, SWD protocol or the bootloader to load the user application into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

In contrast to the ICP method, **in-application programming (IAP)** can use any communication interface supported by the microcontroller (I/Os, USB, CAN, UART, I<sup>2</sup>C, SPI, etc.) to download programming data into memory. IAP allows the user to re-program the Flash memory while the application is running. Nevertheless, part of the application has to have been previously programmed in the Flash memory using ICP.

The program and erase operations are managed through the following seven Flash registers:

- Key register (FLASH\_KEYR)
- Option byte key register (FLASH\_OPTKEYR)
- Flash control register (FLASH\_CR)
- Flash status register (FLASH\_SR)
- Flash address register (FLASH\_AR)
- Option byte register (FLASH\_OBR)
- Write protection register (FLASH\_WRP)

An on going Flash memory operation will not block the CPU as long as the CPU does not access the Flash memory.

On the contrary, during a program/erase operation to the Flash memory, any attempt to read the Flash memory will stall the bus. The read operation will proceed correctly once the program/erase operation has completed. This means that code or data fetches cannot be made while a program/erase operation is ongoing.

For program and erase operations on the Flash memory (write/erase), the internal RC oscillator (HSI) must be ON.

#### Unlocking the Flash memory

After reset, the FPEC (Flash memory program/erase controller) is protected against unwanted write or erase operations. The FLASH\_CR register is not accessible in write mode, except for the OBL\_LAUNCH bit, used to reload the OBL. An unlocking sequence should be written to the FLASH\_KEYR register to open the access to the FLASH\_CR register. This sequence consists of two write operations into FLASH\_KEYR register:

1. Write KEY1 = 0x45670123
2. Write KEY2 = 0xCDEF89AB

Any wrong sequence locks up the FPEC and the FLASH\_CR register until the next reset.

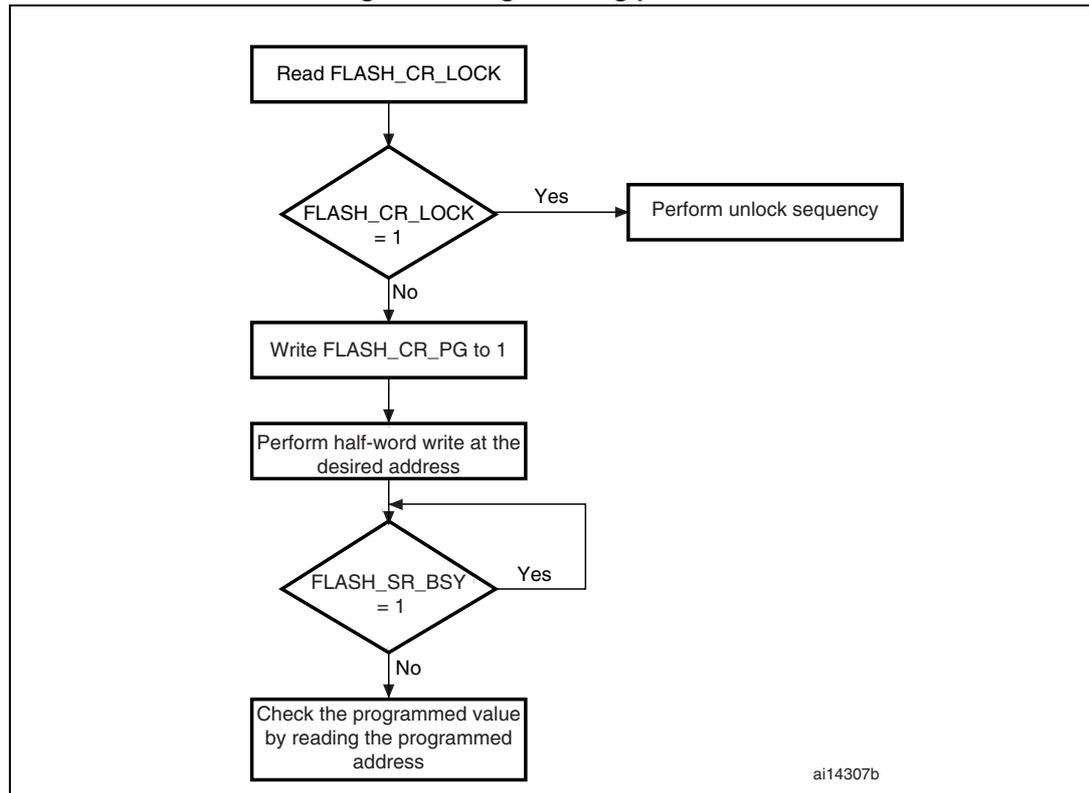
In the case of a wrong key sequence, a bus error is detected and a Hard Fault interrupt is generated. This is done after the first write cycle if KEY1 does not match, or during the second write cycle if KEY1 has been correctly written but KEY2 does not match.

The FPEC and the FLASH\_CR register can be locked again by user software by writing the LOCK bit in the FLASH\_CR register to 1.

**Main Flash memory programming**

The main Flash memory can be programmed 16 bits at a time. The program operation is started when the CPU writes a half-word into a main Flash memory address with the PG bit of the FLASH\_CR register set. Any attempt to write data that are not half-word long will result in a bus error generating a Hard Fault interrupt.

**Figure 2. Programming procedure**



The Flash memory interface preliminarily reads the value at the addressed main Flash memory location and checks that it has been erased. If not, the program operation is skipped and a warning is issued by the PGERR bit in FLASH\_SR register (the only exception to this is when 0x0000 is programmed. In this case, the location is correctly programmed to 0x0000 and the PGERR bit is not set). If the addressed main Flash memory location is write-protected by the FLASH\_WRP register, the program operation is skipped and a warning is issued by the WRPRTERR bit in the FLASH\_SR register. The end of the program operation is indicated by the EOP bit in the FLASH\_SR register.

The main Flash memory programming sequence in standard mode is as follows:

1. Check that no main Flash memory operation is ongoing by checking the BSY bit in the FLASH\_SR register.
2. Set the PG bit in the FLASH\_CR register.
3. Perform the data write (half-word) at the desired address.
4. Wait until the BSY bit is reset in the FLASH\_SR register.
5. Check the EOP flag in the FLASH\_SR register (it is set when the programming operation has succeeded), and then clear it by software.

*Note:* The registers are not accessible in write mode when the BSY bit of the FLASH\_SR register is set.

### Flash memory erase

The Flash memory can be erased page by page or completely (Mass Erase).

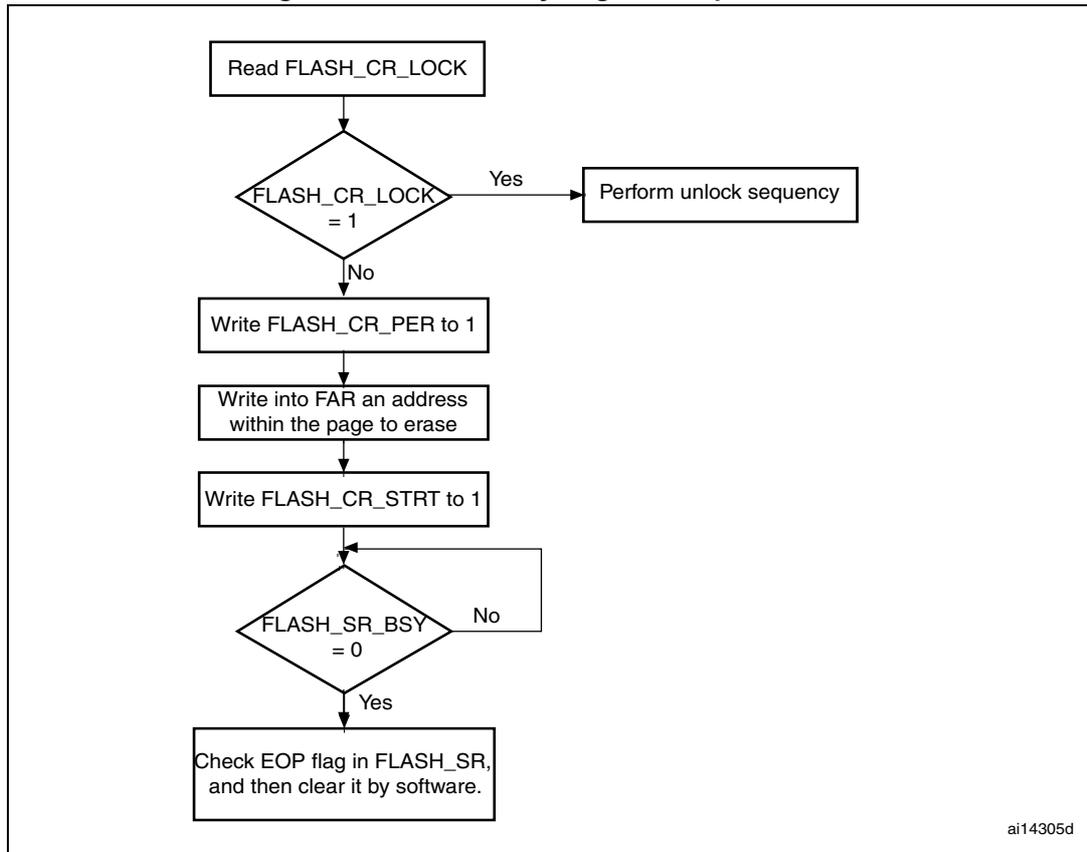
#### Page Erase

To erase a page, the procedure below should be followed:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH\_CR register.
2. Set the PER bit in the FLASH\_CR register
3. Program the FLASH\_AR register to select a page to erase
4. Set the STRT bit in the FLASH\_CR register (see below note)
5. Wait for the BSY bit to be reset
6. Check the EOP flag in the FLASH\_SR register (it is set when the erase operation has succeed), and then clear it by software.
7. Clear the EOP flag.

*Note:* The software should start checking if the BSY bit equals '0' at least one CPU cycle after setting the STRT bit.

Figure 3. Flash memory Page Erase procedure



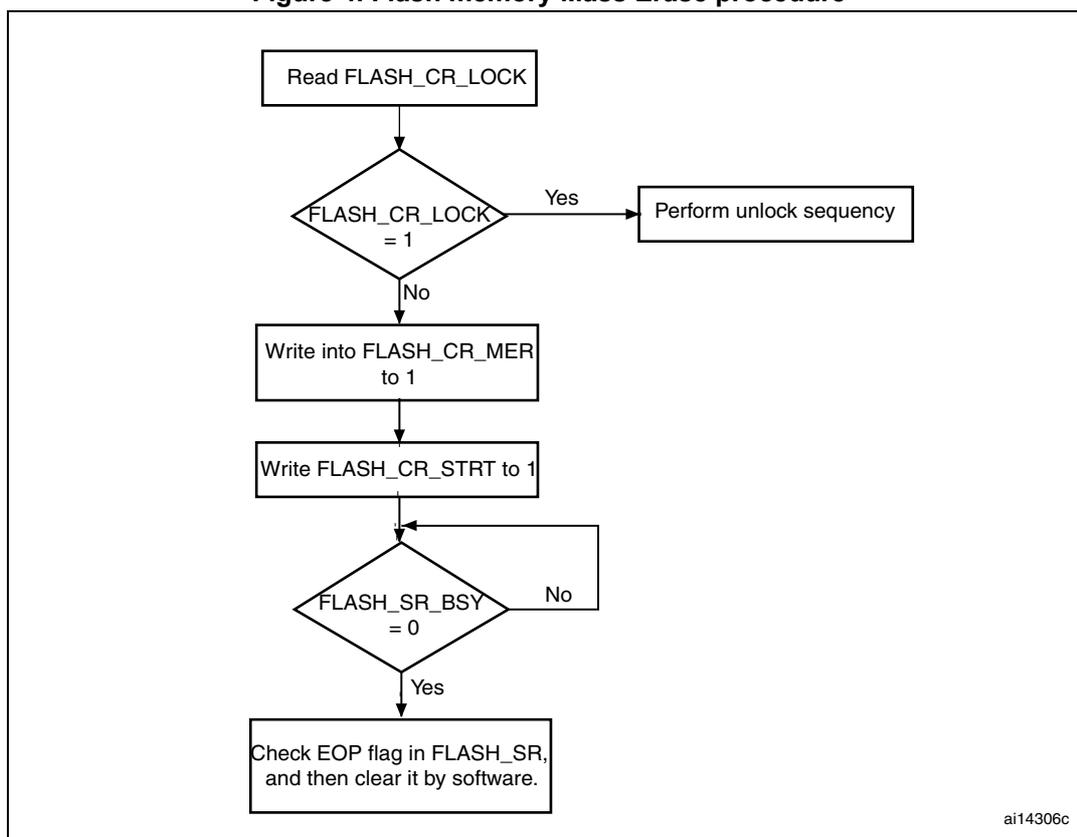
**Mass Erase**

The Mass Erase command can be used to completely erase the user pages of the Flash memory. The information block is unaffected by this procedure. The following sequence is recommended:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH\_SR register
2. Set the MER bit in the FLASH\_CR register
3. Set the STRT bit in the FLASH\_CR register (see below note)
4. Wait for the BSY bit to be reset
5. Check the EOP flag in the FLASH\_SR register (it is set when the erase operation has succeed), and then clear it by software.
6. Clear the EOP flag.

*Note:* The software should start checking if the BSY bit equals '0' at least one CPU cycle after setting the STRT bit.

Figure 4. Flash memory Mass Erase procedure



### Option byte programming

The option bytes are programmed differently from normal user addresses. The number of option bytes is limited to 8 (4 for write protection, 1 for readout protection, 1 for hardware configuration, and 2 for data storage). After unlocking the FPEC, the user has to authorize the programming of the option bytes by writing the same set of KEYS (KEY1 and KEY2) to the FLASH\_OPTKEYR register (refer to [Unlocking the Flash memory](#) for key values). Then, the OPTWRE bit in the FLASH\_CR register will be set by hardware and the user has to set the OPTPG bit in the FLASH\_CR register and perform a half-word write operation at the desired Flash address.

The value of the addressed option byte is first read to check it is really erased. If not, the program operation is skipped and a warning is issued by the WRPRTERR bit in the FLASH\_SR register. The end of the program operation is indicated by the EOP bit in the FLASH\_SR register.

The LSB value is automatically complemented into the MSB before the programming operation starts. This guarantees that the option byte and its complement are always correct.

The sequence is as follows:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the FLASH\_SR register.
2. Unlock the OPTWRE bit in the FLASH\_CR register.
3. Set the OPTPG bit in the FLASH\_CR register
4. Write the data (half-word) to the desired address
5. Wait for the BSY bit to be reset.
6. Read the programmed value and verify.

When the Flash memory read protection option is changed from protected to unprotected, a Mass Erase of the main Flash memory is performed before reprogramming the read protection option. If the user wants to change an option other than the read protection option, then the mass erase is not performed. The erased state of the read protection option byte protects the Flash memory.

### **Erase procedure**

The option byte erase sequence (OPTERASE) is as follows:

1. Check that no Flash memory operation is ongoing by reading the BSY bit in the FLASH\_SR register
2. Unlock the OPTWRE bit in the FLASH\_CR register
3. Set the OPTER bit in the FLASH\_CR register
4. Set the STRT bit in the FLASH\_CR register and wait a few cycles before the BSY bit is set
5. Wait for BSY to reset
6. Read the erased option bytes and verify

### 3.3 Memory protection

The user area of the Flash memory can be protected against read by untrusted code. The pages of the Flash memory can also be protected against unwanted write due to loss of program counter contexts. The write-protection granularity is one sector (four pages).

#### 3.3.1 Read protection

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte.

*Note: If the read protection is set while the debugger is still connected through JTAG/SWD, apply a POR (power-on reset) instead of a system reset.*

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

The Flash memory is protected when the RDP option byte and its complement contain the pair of values shown in [Table 4](#).

**Table 4. Flash memory read protection status**

RDP byte value	RDP complement value	Read protection level
0xAA	0x55	Level 0 (ST production configuration)
Any value except 0xAA or 0xCC	Any value (not necessarily complementary) except 0x55 and 0x33	Level 1
0xCC	0x33	Level 2

The System memory area is read accessible whatever the protection level. It is never accessible for program/erase operation

#### Level 0: no protection

Read, program and erase operations into the main memory Flash area are possible. The option bytes are also accessible by all operations.

### Level 1: Read protection

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

- **User mode:** Code executing in user mode can access main memory Flash and option bytes with all operations.
- **Debug, boot RAM and bootloader modes:** In debug mode or when code is running from boot RAM or bootloader, the main Flash memory and the backup registers (RTC\_BKPxR in the RTC) are totally inaccessible. In these modes, even a simple read access generates a bus error and a Hard Fault interrupt. The main memory is program/erase protected to prevent malicious or unauthorized users from reprogramming any of the user code with a dump routine. Any attempted program/erase operation sets the PGERR flag of Flash status register (FLASH\_SR). When the RDP is reprogrammed to the value 0xAA to move back to Level 0, a mass erase of main memory Flash is performed and the backup registers (RTC\_BKPxR in the RTC) are reset.

### Level 2: No debug

In this level, the protection level 1 is guaranteed. In addition, the Cortex<sup>®</sup>-M4 with FPU debug capabilities are disabled. Consequently, the debug port, the boot from RAM (boot RAM mode) and the boot from System memory (bootloader mode) are no more available. In user execution mode, all operations are allowed on the Main Flash memory. On the contrary, only read and program operations can be performed on the option bytes.

Option bytes cannot be erased. Moreover, the RDP bytes cannot be programmed. Thus, the level 2 cannot be removed at all: it is an irreversible operation. When attempting to program the RDP byte, the protection error flag WRPRERR is set in the FLASH\_SR register and an interrupt can be generated.

*Note:* The debug feature is also disabled under reset.

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

**Table 5. Access status versus protection level and execution modes**

Area	Read protection level (RDP)	User execution			Debug/ BootFromRam/ BootFromLoader		
		Read	Write	Erase	Read	Write	Erase
Main Flash memory	1	Yes	Yes	Yes	No	No	No <sup>(3)</sup>
	2	Yes	Yes	Yes	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
System memory <sup>(2)</sup>	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	NA <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
Option bytes	1	Yes	Yes <sup>(3)</sup>	Yes	No	No	Yes
	2	Yes	Yes <sup>(4)</sup>	No	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
Backup registers	1	Yes	Yes	N/A	No	No	No <sup>(5)</sup>
	2	Yes	Yes	N/A	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>

1. When the protection level 2 is active, the Debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3. The main Flash memory is erased when the RDP option byte is programmed with all level protections disabled (0xAA).
4. All option bytes can be programmed, except the RDP byte.
5. The backup registers are erased only when RDP changes from level 1 to level 0.

### Changing read protection level

It is easy to move from level 0 to level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from level 0 or from level 1. On the contrary, the change to level 0 (no protection) is not possible without a main Flash memory Mass Erase operation. This Mass Erase is generated as soon as 0xAA is programmed in the RDP byte.

*Note: When the Mass Erase command is used, the backup registers (RTC\_BKPxR in the RTC) are also reset.*

*To validate the protection level change, the option bytes must be reloaded through the OBL\_LAUNCH bit in Flash control register.*

### 3.3.2 Write protection

The write protection is implemented with a granularity of 2 pages. It is activated by configuring the WRP[1:0] option bytes, and then by reloading them by setting the OBL\_LAUNCH bit in the FLASH\_CR register.

If a program or an erase operation is performed on a protected sector, the Flash memory returns a WRPRERR protection error flag in the Flash memory Status Register (FLASH\_SR).

### Write unprotection

To disable the write protection, two application cases are provided:

- Case 1: Read protection disabled after the write unprotection:
  - Erase the entire option byte area by using the OPTER bit in the Flash memory control register (FLASH\_CR).
  - Program the code 0xAA in the RDP byte to unprotect the memory. This operation forces a Mass Erase of the main Flash memory.
  - Set the OBL\_LAUNCH bit in the Flash control register (FLASH\_CR) to reload the option bytes (and the new WRP[3:0] bytes), and to disable the write protection.
- Case 2: Read protection maintained active after the write unprotection, useful for in-application programming with a user bootloader:
  - Erase the entire option byte area by using the OPTER bit in the Flash memory control register (FLASH\_CR).
  - Set the OBL\_LAUNCH bit in the Flash control register (FLASH\_CR) to reload the option bytes (and the new WRP[3:0] bytes), and to disable the write protection.

### 3.3.3 Option byte block write protection

The option bytes are always read-accessible and write-protected by default. To gain write access (Program/Erase) to the option bytes, a sequence of keys (same as for lock) has to be written into the OPTKEYR. A correct sequence of keys gives write access to the option bytes and this is indicated by OPTWRE in the FLASH\_CR register being set. Write access can be disabled by resetting the bit through software.

## 3.4 Flash interrupts

Table 6. Flash interrupt request

Interrupt event	Event flag	Enable control bit
End of operation	EOP	EOPIE
Write protection error	WRPRERR	ERRIE
Programming error	PGERR	ERRIE

### 3.5 Flash register description

The Flash memory registers have to be accessed by 32-bit words (half-word and byte accesses are not allowed).

#### 3.5.1 Flash access control register (FLASH\_ACR)

Address offset: 0x00

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PRFT BS	PRFT BE	Res.	LATENCY[2:0]											
										r	rW		rW	rW	rW

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

Bit 5 **PRFTBS**: Prefetch buffer status

This bit provides the status of the prefetch buffer.

0: Prefetch buffer is disabled

1: Prefetch buffer is enabled

Bit 4 **PRFTBE**: Prefetch buffer enable

0: Prefetch is disabled

1: Prefetch is enabled

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **LATENCY[2:0]**: Latency

These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.

000: Zero wait state, if  $0 < \text{SYSCLK} \leq 24 \text{ MHz}$

001: One wait state, if  $24 \text{ MHz} < \text{SYSCLK} \leq 48 \text{ MHz}$

010: Two wait states, if  $48 < \text{SYSCLK} \leq 72 \text{ MHz}$

#### 3.5.2 Flash key register (FLASH\_KEYR)

Address offset: 0x04

Reset value: xxxx xxxx

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

*Note:* These bits are all write-only and return a 0 when read.

Bits 31:0 **FKEYR**: Flash key

These bits represent the keys to unlock the Flash.



### 3.5.3 Flash option key register (FLASH\_OPTKEYR)

Address offset: 0x08  
 Reset value: xxxx xxxx

All the register bits are write-only and return a 0 when read.

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

Bits 31:0 **OPTKEYR**: Option byte key  
 These bits represent the keys to unlock the OPTWRE.

### 3.5.4 Flash status register (FLASH\_SR)

Address offset: 0x0C  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EOP	WRPRT ERR	Res.	PG ERR	Res.	BSY									
										rw	rw		rw		r

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

Bit 5 **EOP**: End of operation

Set by hardware when a Flash operation (programming / erase) is completed.  
 Reset by writing a 1

*Note: EOP is asserted at the end of each successful program or erase operation*

Bit 4 **WRPRTERR**: Write protection error

Set by hardware when programming a write-protected address of the Flash memory.  
 Reset by writing 1.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PGERR**: Programming error

Set by hardware when an address to be programmed contains a value different from '0xFFFF' before programming.  
 Reset by writing 1.

*Note: The STRT bit in the FLASH\_CR register should be reset before starting a programming operation.*

Bit 1 Reserved, must be kept at reset value

Bit 0 **BSY**: Busy

This indicates that a Flash operation is in progress. This is set on the beginning of a Flash operation and reset when the operation finishes or when an error occurs.

### 3.5.5 Flash control register (FLASH\_CR)

Address offset: 0x10

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OBL_L AUNCH	EOPIE	Res.	ERRIE	OPTWR E	Res.	LOCK	STRT	OPTER	OPT PG	Res.	MER	PER	PG
		rW	rW		rW	rW		rW	rW	rW	rW		rW	rW	rW

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

Bit 13 **OBL\_LAUNCH**: Force option byte loading

When set to 1, this bit forces the option byte reloading. This operation generates a system reset.

- 0: Inactive
- 1: Active

Bit 12 **EOPIE**: End of operation interrupt enable

This bit enables the interrupt generation when the EOP bit in the FLASH\_SR register goes to 1.

- 0: Interrupt generation disabled
- 1: Interrupt generation enabled

Bit 11 Reserved, must be kept at reset value

Bit 10 **ERRIE**: Error interrupt enable

This bit enables the interrupt generation on an error when PGERR / WRPRERR are set in the FLASH\_SR register.

- 0: Interrupt generation disabled
- 1: Interrupt generation enabled

Bit 9 **OPTWRE**: Option bytes write enable

When set, the option bytes can be programmed. This bit is set on writing the correct key sequence to the FLASH\_OPTKEYR register.

This bit can be reset by software

Bit 8 Reserved, must be kept at reset value.

Bit 7 **LOCK**: Lock

Write to 1 only. When it is set, it indicates that the Flash is locked. This bit is reset by hardware after detecting the unlock sequence.

In the event of unsuccessful unlock operation, this bit remains set until the next reset.

Bit 6 **STRT**: Start

This bit triggers an ERASE operation when set. This bit is set only by software and reset when the BSY bit is reset.

Bit 5 **OPTER**: Option byte erase

Option byte erase chosen.

Bit 4 **OPTPG**: Option byte programming

Option byte programming chosen.

- Bit 3 Reserved, must be kept at reset value.
- Bit 2 **MER**: Mass erase  
Erase of all user pages chosen.
- Bit 1 **PER**: Page erase  
Page Erase chosen.
- Bit 0 **PG**: Programming  
Flash programming chosen.

### 3.5.6 Flash address register (FLASH\_AR)

Address offset: 0x14

Reset value: 0x0000 0000

This register is updated by hardware with the currently/last used address. For Page Erase operations, this should be updated by software to indicate the chosen page.

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

Bits 31:0 **FAR**: Flash Address

Chooses the address to program when programming is selected, or a page to erase when Page Erase is selected.

*Note: Write access to this register is blocked when the BSY bit in the FLASH\_SR register is set.*

### 3.5.7 Option byte register (FLASH\_OBR)

Address offset 0x1C

Reset value: 0xFFFFFFFF

It contains the level protection notifications, error during load of option bytes and user options.

The reset value of this register depends on the value programmed in the option byte and the OPTERR bit reset value depends on the comparison of the option byte and its complement during the option byte loading phase.

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
Data1								Data0								SDADC12_VDD_MONITOR	SRAM_PE	VDDA_MONITOR	nBOOT1	Res.	nRST_STDBY	nRST_STOP	WDG_SW	Res.					LEVEL2_PROT	LEVEL1_PROT	OPTERR
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Data1

Bits 23:16 Data0

Bits 15:8 **OBR**: User Option Byte

Bit 15: SDADC12\_VDD\_MONITOR

Bit 14: SRAM\_PE

Bit 13: VDDA\_MONITOR

Bit 12: nBOOT1

Bit 11: Reserved, must be kept at reset value.

Bit 10: nRST\_STDBY

Bit 9: nRST\_STOP

Bit 8: WDG\_SW

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

Bit 2 **LEVEL2\_PROT**: Level 2 Read protection status

When set, this indicates that the Flash memory is level 2 read-protected. It is set only if LEVEL1\_PROT is set.

*Note: This bit is read-only.*

Bit 1 **LEVEL1\_PROT**: Level 1 Read protection status

When set, this indicates that the Flash memory is level 1 read-protected.

*Note: This bit is read-only.*

Bit 0 **OPTERR**: Option byte Load error

When set, this indicates that the loaded option byte and its complement do not match. The corresponding byte and its complement are read as 0xFF in the FLASH\_OBR or FLASH\_WRPR register.

*Note: This bit is read-only.*

### 3.5.8 Write protection register (FLASH\_WRP)

Address offset: 0x20

Reset value: 0xFFFF FFFF

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

Bits 31:0 **WRP**: Write protect

This register contains the write-protection option bytes loaded by the OBL.  
These bits are read-only.

### 3.6 Flash register map

Table 7. Flash interface - 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	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTBS	PRFTBE	Res.	LATENCY [2:0]				
	Reset value																											1	1		0	0	0	
0x004	FLASH_KEYR	FKEYR[31:0]																																
	Reset value	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	x	x	x	x	x
0x008	FLASH_OPTKEYR	OPTKEYR[31:0]																																
	Reset Value	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	x	x	x	x	x
0x00C	FLASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOP	WRPTERR	Res.	PGERR	Res.	BSY
	Reset value																												0	0	0	0	0	0
0x010	FLASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBL_LAUNCH	EOPIE	Res.	ERRIE	OPTWRE	Res.	LOCK	STRT	OPTER	OPTPG	Res.	MER	PER	PG
	Reset value																				0	0		0	0		1	0	0	0	0	0	0	0
0x014	FLASH_AR	FAR[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 7. Flash interface - 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	
0x01C	FLASH_OBR	Data1								Data0								SDADC12_VDD_MONITOR	SRAM_PE	VDDA_MONITOR	nBOOT1	Res.	nRST_STDBY	nRST_STOP	WDG_SW	Res.						LEVEL2_PROT	LEVEL1_PROT	OPTERR
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	1						0	1	0
0x020	FLASH_WRP	WRP[31:0]																																
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to [Section 2.2.2 on page 39](#) for the register boundary addresses.

## 4 Option byte description

There are eight option bytes. They are configured by the end user depending on the application requirements. As a configuration example, the watchdog may be selected in hardware or software mode.

A 32-bit word is split up as follows in the option bytes.

**Table 8. Option byte format**

31-24	23-16	15 -8	7-0
Complemented option byte 1	Option byte 1	Complemented option byte 0	Option byte 0

The organization of these bytes inside the information block is as shown in [Table 9](#).

The option bytes can be read from the memory locations listed in [Table 9](#) or from the Option byte register (FLASH\_OBR).

*Note:* The new programmed option bytes (user, read/write protection) are loaded after a system reset.

**Table 9. Option byte organization**

Address	[31:24]	[23:16]	[15:8]	[7:0]
0x1FFF F800	nUSER	USER	nRDP	RDP
0x1FFF F804	nData1	Data1	nData0	Data0
0x1FFF F808	nWRP1	WRP1	nWRP0	WRP0
0x1FFF F80C	nWRP3	WRP3	nWRP2	WRP2

Table 10. Description of the option bytes

Flash memory address	Option bytes
0x1FFF F800	<p>Bits [31:24] <b>nUSER</b></p> <p>Bits [23:16] <b>USER</b>: User option byte (stored in FLASH_OBR[15:8])                      This byte is used to configure the following features:</p> <ul style="list-style-type: none"> <li>- Select the watchdog event: Hardware or software</li> <li>- Reset event when entering Stop mode</li> <li>- Reset event when entering Standby mode</li> </ul> <p>Bit 23: <b>SDADC12_VDD MONITOR</b></p> <ul style="list-style-type: none"> <li>0: SDADC12_VDD power supply supervisor disabled.</li> <li>1: SDADC12_VDD power supply supervisor enabled</li> </ul> <p>Bit 22: <b>SRAM_PE</b></p> <p>The SRAM hardware parity check is disabled by default. This bit allows the user to enable the SRAM hardware parity check.</p> <ul style="list-style-type: none"> <li>0: Parity check enabled.</li> <li>1: Parity check disabled.</li> </ul> <p>Bit 21: <b>VDDA_MONITOR</b></p> <p>This bit selects the analog monitoring on the VDDA power source:</p> <ul style="list-style-type: none"> <li>0: VDDA power supply supervisor disabled.</li> <li>1: VDDA power supply supervisor enabled.</li> </ul> <p>Bit 20: <b>nBOOT1</b></p> <p>Together with the BOOT0 pin, this bit selects Boot mode from the main Flash memory, SRAM or System memory. Refer to <a href="#">Section 2.6: Boot configuration</a>.</p> <p>Bit 19: Reserved, must be kept at reset.</p> <p>Bit 18: <b>nRST_STDBY</b></p> <ul style="list-style-type: none"> <li>0: Reset generated when entering Standby mode.</li> <li>1: No reset generated.</li> </ul> <p>Bit 17: <b>nRST_STOP</b></p> <ul style="list-style-type: none"> <li>0: Reset generated when entering Stop mode</li> <li>1: No reset generated</li> </ul> <p>Bit 16: <b>WDG_SW</b></p> <ul style="list-style-type: none"> <li>0: Hardware watchdog</li> <li>1: Software watchdog</li> </ul> <p>Bits [15:8]: <b>nRDP</b></p> <p>Bits [7:0]: <b>RDP</b>: Read protection option byte</p> <p>The value of this byte defines the Flash memory protection level</p> <ul style="list-style-type: none"> <li>0xAA: Level 0</li> <li>0xFF (except 0xAA and 0xCC): Level 1</li> <li>0xCC: Level 2</li> </ul> <p>The protection levels 1 and 2 are stored in the Flash_OBR Flash option register (LEVEL1_PROT and LEVEL2_PROT status flags respectively).</p>
0x1FFF F804	<p><b>Datax</b>: Two bytes for user data storage.</p> <p>These addresses can be programmed using the option byte programming procedure.</p> <p>Bits [31:24]: <b>nData1</b></p> <p>Bits [23:16]: <b>Data1</b> (stored in FLASH_OBR[31:24])</p> <p>Bits [15:8]: <b>nData0</b></p> <p>Bits [7:0]: <b>Data0</b> (stored in FLASH_OBR[23:16])</p>

Table 10. Description of the option bytes (continued)

Flash memory address	Option bytes
0x1FFF F808	<p><b>WRP<sub>x</sub></b>: Flash memory write protection option bytes</p> <p>Bits [31:24]: <b>nWRP1</b></p> <p>Bits [23:16]: <b>WRP1</b> (stored in FLASH_WRP[15:8])</p> <p>Bits [15:8]: <b>nWRP0</b></p> <p>Bits [7:0]: <b>WRP0</b> (stored in FLASH_WRP[7:0])</p> <p>0: Write protection active 1</p> <p>0: Write protection not active</p> <p>Refer to <a href="#">Section 3.3.2: Write protection</a> for more details.</p>
0x1FFF F80C	<p><b>WRP<sub>x</sub></b>: Flash memory write protection option bytes</p> <p>Bits [31:24]: <b>nWRP3</b></p> <p>Bits [23:16]: <b>WRP3</b> (stored in FLASH_WRP[31:24])</p> <p>Bits [15:8]: <b>nWRP2</b></p> <p>Bits [7:0]: <b>WRP2</b> (stored in FLASH_WRP[23:16])</p> <p>One bit of the user option bytes WRP<sub>x</sub> is used to protect 2 pages of 2 Kbytes in the main memory block.</p> <p>0: Write protection active</p> <p>1: Write protection not active</p> <p>In total, 4 user option bytes are used to protect the whole main Flash memory.</p> <p>WRP0: Write-protects pages 0 to 15</p> <p>WRP1: Write-protects pages 16 to 31</p> <p>WRP2: Write-protects pages 32 to 47</p> <p>WRP3: bits 0-6 write-protect pages 48 to 61, bit 7 write-protects pages 62 to 127.</p>

On every system reset, the option byte loader (OBL) reads the information block and stores the data into the Option byte register (FLASH\_OBR) and the Write protection register (FLASH\_WRP). Each option byte also has its complement in the information block. During option loading, by verifying the option bit and its complement, it is possible to check that the loading has correctly taken place. If this is not the case, an option byte error (OPTERR) is generated. When a comparison error occurs, the corresponding option byte is forced to 0xFF. The comparator is disabled when the option byte and its complement are both equal to 0xFF (Electrical Erase state).

## 5 Cyclic redundancy check calculation unit (CRC)

### 5.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

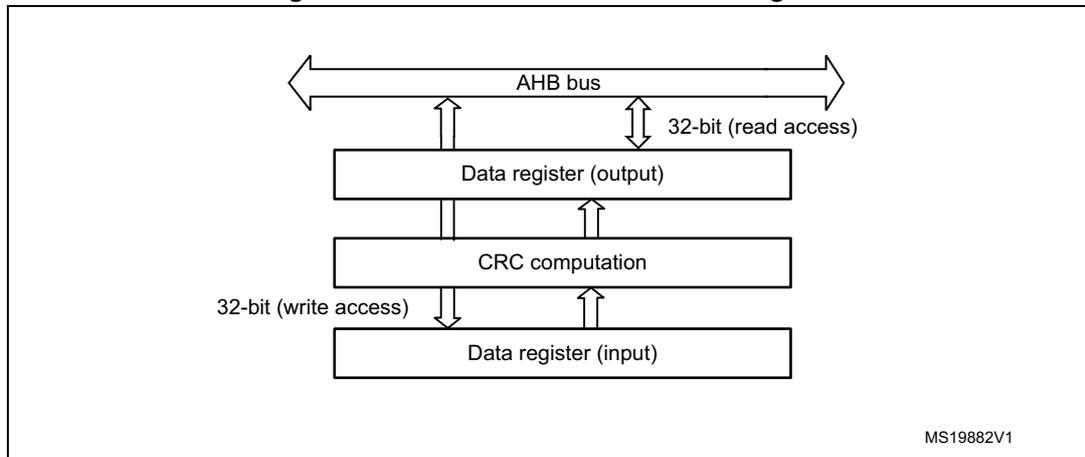
Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to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

### 5.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial:  $0x4C11DB7$   
$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$
- Alternatively uses a fully programmable polynomial with programmable size (7, 8, 16, 32 bits).
- Handles 8-, 16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data

## 5.3 CRC functional description

Figure 5. CRC calculation unit block diagram



The CRC calculation unit has a single 32-bit read/write data register (CRC\_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC\_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC\_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:

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

An input buffer allows to immediately write a second data without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV\_IN[1:0] bits in the CRC\_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV\_OUT bit in the CRC\_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC\_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC\_INIT register. The CRC\_DR register is automatically initialized upon CRC\_INIT register write access.

The CRC\_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC\_CR register.

**Polynomial programmability**

The polynomial coefficients are fully programmable through the CRC\_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC\_CR register.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC\_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC\_DR read before changing the polynomial.

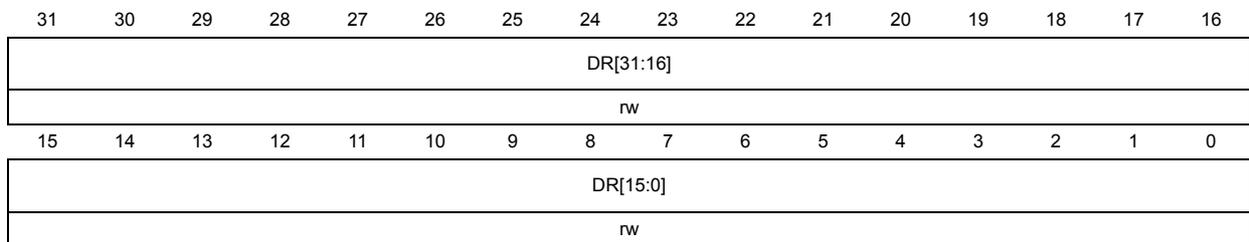
The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

**5.4 CRC registers**

**5.4.1 Data register (CRC\_DR)**

Address offset: 0x00

Reset value: 0xFFFF FFFF



Bits 31:0 **DR[31:0]**: Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

### 5.4.2 Independent data register (CRC\_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept cleared.

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

These 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

### 5.4.3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET							
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept cleared.

Bit 7 **REV\_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV\_IN[1:0]**: Reverse input data

These bits control the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept cleared.

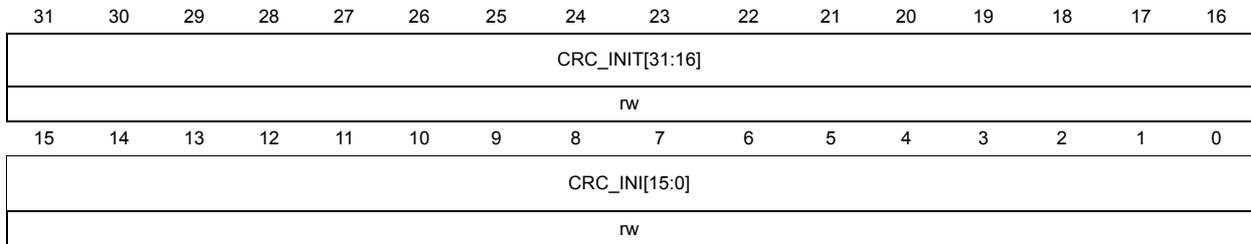
Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC\_INIT register. This bit can only be set, it is automatically cleared by hardware

### 5.4.4 Initial CRC value (CRC\_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF



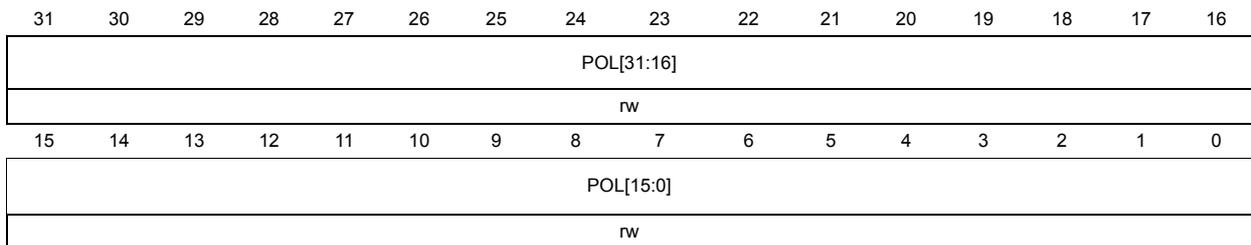
Bits 31:0 **CRC\_INIT**: Programmable initial CRC value

This register is used to write the CRC initial value.

### 5.4.5 CRC polynomial (CRC\_POL)

Address offset: 0x14

Reset value: 0x04C11DB7



Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32-bits, the least significant bits have to be used to program the correct value.

5.4.6 CRC register map

Table 11. CRC 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	CRC_DR	DR[31:0]																																			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
0x04	CRC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]										
	Reset value																										0	0	0	0	0	0	0	0	0		
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																										0	0	0	0	0	0	0	0	0		
0x10	CRC_INIT	CRC_INIT[31:0]																																			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
0x14	CRC_POL	Polynomial coefficients																																			
	Reset value	0x04C11DB7																																			

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 6 Power control (PWR)

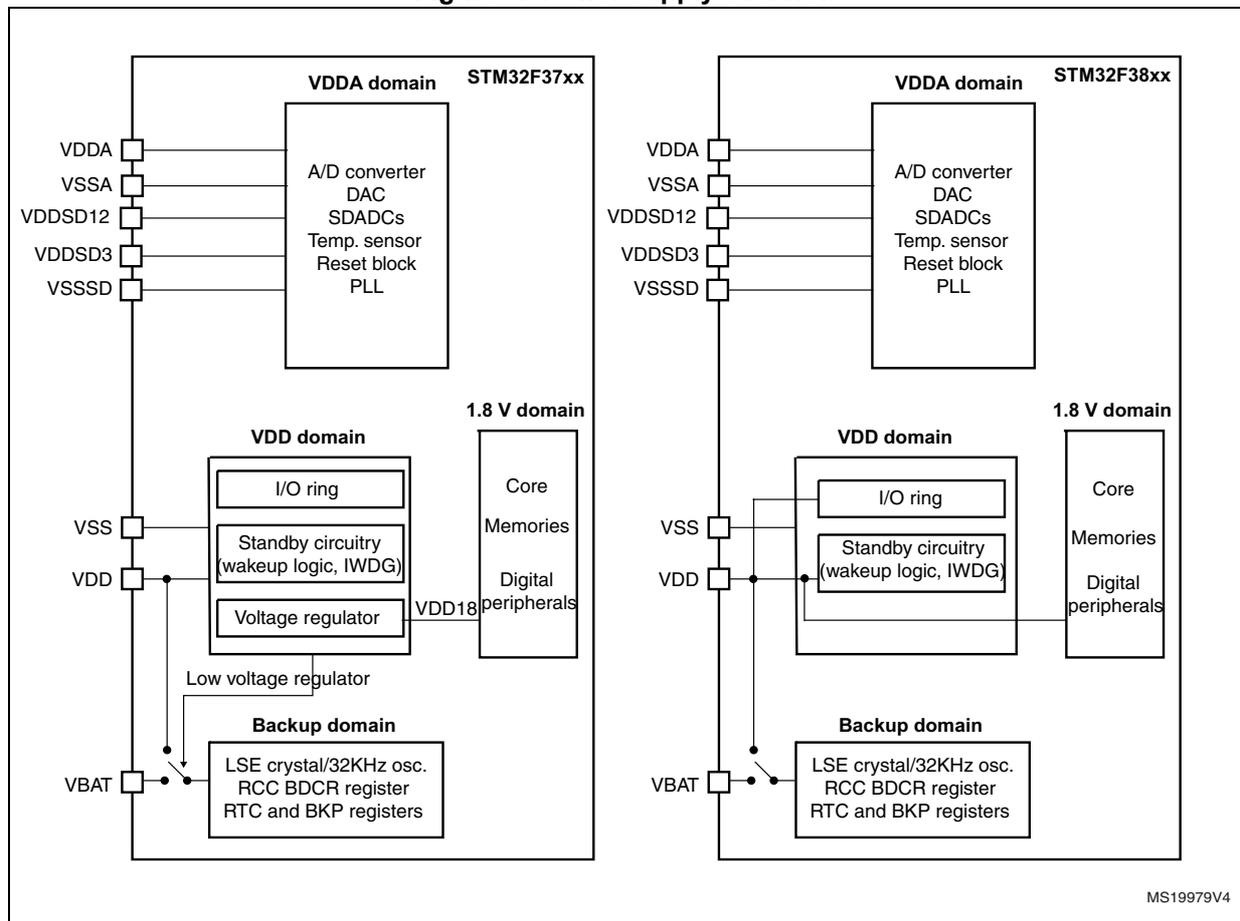
### 6.1 Power supplies

An internal regulator is embedded in the STM32F373xx devices.

- The internal regulator is enabled in the STM32F373xx MCUs:  
The STM32F373xx devices require 2.0 V - 3.6 V operating supply voltage ( $V_{DD}$ ) and 2.0 V - 3.6 V analog supply voltage ( $V_{DDA}$ ). The embedded regulator is used to supply the internal 1.8 V digital power.
- The internal regulator is disabled in the STM32F378xx MCUs:  
The STM32F378xx devices require 1.8 V +/- 8 % operating voltage supply ( $V_{DD}$ ) and 1.65 V - 3.6 V analog voltage supply ( $V_{DDA}$ ). The embedded regulator is OFF and  $V_{DD}$  directly supplies the regulator output.

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 6. Power supply overview



## Supply voltages

- $V_{DD}$ ,  $V_{SS}$  = 2.0 to 3.6 V: external power supply for I/Os (except I/Os related to SDADCs) and core.

These supply voltages are provided externally through  $V_{DD}$  and  $V_{SS}$  pins.

$V_{DD}$  = 2.0 to 3.6 V (STM32F373xx devices) or  $1.8\text{ V} \pm 8\%$  (STM32F378xx devices).

In case of the 1.8 V mode external supply,  $V_{DD}$  directly supplies the regulator output which directly drives the  $V_{DD18}$  domain.  $V_{DD}$  must always be kept lower or equal to  $V_{DDA}$ .
- $V_{DD18}$  = 1.65 to 1.95 V ( $V_{DD18}$  domain): power supply for digital core, SRAM and Flash memory.

$V_{DD18}$  is either internally generated through an internal voltage regulator (STM32F373xx devices) or can be provided directly from the external  $V_{DD}$  pin when the regulator is bypassed (STM32F378xx devices).
- $V_{DDA}$ ,  $V_{SSA}$  = 2.4 to 3.6 V (ADC/DAC ON) or 1.65 to 3.6 V (both ADC/DAC OFF): external power supply for ADC, DAC, comparators, temperature sensor, PLL, HSI 8 MHz oscillator, and LSI 40 kHz oscillator.

It is forbidden to have  $V_{DDA} < V_{DD} - 0.4\text{ V}$ . A external Schottky diode must be placed between  $V_{DD}$  and  $V_{DDA}$  to guarantee this condition is met.
- $V_{BAT}$  = 1.65 to 3.6 V

Backup power supply for RTC, LSE oscillator, PC13 to PC15 and backup registers when  $V_{DD}$  is not present. When  $V_{DD}$  supply is present, the internal power switch switches the backup power to  $V_{DD}$ .

During fast  $V_{DD}$  startup, if  $V_{DD} + 0.6 > V_{BAT}$ , there is a 2 ms period of time during which the current can flow from  $V_{DD}$  to  $V_{BAT}$  pin.

In STM32F378xx devices,  $V_{BAT}$  must be connected to  $V_{DD}$  (no battery backup).

If  $V_{BAT}$  is not used, it must be connected to  $V_{DD}$ .
- $V_{DDSD12}$  = 2.2 to 3.6 V: external power supply for SDADC1/2, PB2, PB10, and PE7 to PE15 I/O pins (I/O pin ground is internally connected to  $V_{SS}$ ).

$V_{DDSD12}$  must always be kept lower or equal to  $V_{DDA}$ . If  $V_{DDSD12}$  is not used, it must be connected to  $V_{DDA}$ .
- $V_{DDSD3}$  = 2.2 to 3.6 V: external power supply for SDADC3, PB14 to PB15 and PD8 to PD15 I/O pins (I/O pin ground is internally connected to  $V_{SS}$ ).

$V_{DDSD3}$  must always be kept lower or equal to  $V_{DDA}$ . If  $V_{DDSD3}$  is not used, it must be connected to  $V_{DDA}$ .
- $V_{SSSD}$ : analog ground pin for SDADC1/2/3.

$V_{SSSD}$  must be connected to ground.

**Note:** *PB0 and PB1 pins are powered from  $V_{DD}$  power supply. However, PB0 and PB1 are also sharing SDADC1 analog inputs. Therefore the maximum voltage connected to these pins when they are not used as analog inputs must be less than the minimum of  $V_{DD}$  and  $V_{DDSD12}$  supply voltages to avoid current injection into  $V_{DD}$  and  $V_{DDSD12}$ .*

*When PB0 and PB1 are configured in analog input mode ( $MODERy[1:0] = 11$ , see [Section 8.4.1: GPIO port mode register \(GPIOx\\_MODER\) \(x = A..F\)](#)), the maximum voltage must be less than  $V_{DDSD12}$ .*

*If  $V_{DD}$  is higher than  $V_{DDSD12}$ , it is forbidden to use PB0 and PB1 in digital output mode to avoid current injection from  $V_{DD}$  supply into  $V_{DDSD12}$  supply through shared analog inputs.*

### Reference voltages

- $V_{REF+}$ ,  $V_{REF-}$  = 2.2 to 3.6 V  
 $V_{REF+}$  and  $V_{REF-}$  correspond to the reference voltage for ADC and DAC peripherals. They define the ADC and DAC input range.  
 $V_{REF+}$  must always be kept lower or equal to  $V_{DDA}$ . If  $V_{REF+}$  is not used, it must be connected to  $V_{DDA}$  and  $V_{REF-}$  must be connected to  $V_{SSA}$ .  
 $V_{REF-}$  must be connected to ground.
- $V_{REFSD+}$ ,  $V_{REFSD-}$  = 1.1 to 3.6 V  
 $V_{REFSD+}$  and  $V_{REFSD-}$  correspond to the reference voltage for SDADCx converters. They define the input conversion range for all SDADCx converters.  
 If SDADCx is configured in external reference voltage mode, the external voltage reference source must be connected to these pins.  
 If the external reference voltage is not enabled, then the selected SDADC internal reference voltage source ( $V_{REFINT}$ ,  $SDADC\_V_{DD}$ ) is present on  $V_{REFSD+}$ .  
 A 10 nF+1  $\mu$ F capacitor must be placed between  $V_{REFSD+}$  and  $V_{REFSD-}$  for decoupling purposes.  
 $V_{REFSD+}$  must be lower than SDADC power supply:  

$$V_{REFSD+} < \min(V_{DDSD12}, V_{DDSD3}).$$
 $V_{REFSD-}$  must be connected to ground.

#### 6.1.1 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  $V_{SSA}$ .
- The SDADC voltage supply is available on separate  $V_{DDSDx}$  pins.
- An isolated SDADC supply ground connection is provided on pin  $V_{SSSD}$ .

The  $V_{DDA}$  supply/reference voltage can be equal or higher than  $V_{DD}$ .

The  $V_{DDSD12}$  and  $V_{DDSD3}$  can be different from  $V_{DD}$ ,  $V_{DDA}$  and from one another, considering they are inside the allowed working range and must always be lower than  $V_{DDA}$ . When  $V_{DDSD3}$  is different, it must start before or at the same time as  $V_{DDSD12}$ .

When a single supply is used,  $V_{DDA}$ ,  $V_{DDSD12}$  and  $V_{DDSD3}$  can be externally connected to  $V_{DD}$ , through the external filtering circuit in order to ensure a noise free analog supply/reference voltage.

When  $V_{DDA}$  is different from  $V_{DD}$ , it must always be higher or equal to  $V_{DD}$ . In order to ensure this condition, also during power-up/power-down transitions, an external Schottky diode may be used between  $V_{DD}$  and  $V_{DDA}$ .

$V_{DDSD12}$  and  $V_{DDSD3}$  can also be different, higher or lower than  $V_{DD}$  and lower or equal to  $V_{DDA}$ .

## 6.1.2 Battery backup domain

To retain the content of the backup registers and supply the whole RTC domain 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 battery backup feature is not available on STM32F378xx microcontrollers ( $V_{DD} = 1.8 \text{ V} \pm 8\%$ ). When the device operates in this mode,  $V_{BAT}$  pin must be connected to  $V_{DD}$ .

The  $V_{BAT}$  pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 I/Os, 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  $V_{DD} > V_{BAT} + 0.6 \text{ V}$ , 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 AN4206).

When the RTC 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 23.6.19: RTC backup registers \(RTC\\_BKPxR\) on page 548](#))

*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 I/Os must not be used as a current source (e.g. to drive an LED).

When the RTC 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 23.6.15: RTC calibration register \(RTC\\_CALR\) on page 542](#))

### 6.1.3 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. This mode is automatically disabled when the USART, CEC, or I2C peripheral requires a clock in Stop mode.
- In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost except for the Standby circuitry and the RTC Domain.

#### STM32F378xx microcontrollers

The internal voltage regulator is bypassed in STM32F378xx devices (unlike the STM32F373xx where the voltage regulator is functional). In this case, the microcontroller must be powered from a nominal  $V_{DD} = 1.8 \text{ V} \pm 8\%$  voltage.

In STM32F378xx microcontrollers, the external NPOR input pin replaces the internal POR signal. The external NPOR pin must be controlled by the application (released after all supply voltages are stabilized) and be connected to  $V_{DDA}$  through a pull-up resistor. This pin replaces PB2 GPIO pin.

## 6.2 Power supply supervisor

### 6.2.1 Power on reset (POR)/power down reset (PDR)

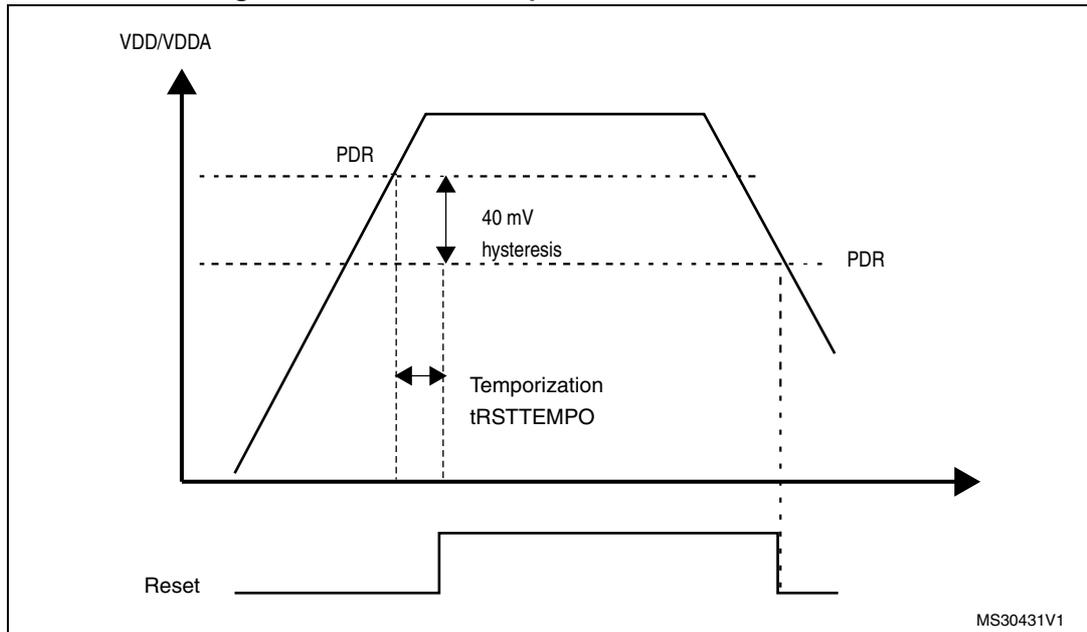
The device has an integrated power-on reset (POR) and power-down reset (PDR) circuits which are always active and ensure proper operation above a threshold of 2 V.

The device remains in Reset mode when the monitored supply voltage is below a specified threshold,  $V_{POR/PDR}$ , without the need for an external reset circuit.

- The POR monitors only the  $V_{DD}/V_{DDSD12}$  supply voltage. During startup phase,  $V_{DDA}$  must arrive first and be higher than or equal to  $V_{DD}$ .
- The PDR monitors all  $V_{DD}/V_{DDA}/V_{DDSD12}$  supply voltages. However, if the application is designed with  $V_{DDA}/V_{DDSD12}$  higher than or equal to  $V_{DD}$ , the  $V_{DDA}$  and  $V_{DDSD12}$  power supply supervisor can be disabled (by programming a dedicated  $V_{DDA\_MONITOR}$  and  $SDADC12\_V_{DD\_MONITOR}$  option bits) to reduce the power consumption.

For more details on the power on /power down reset threshold, refer to the electrical characteristics section in the datasheet.

Figure 7. Power on reset/power down reset waveform



### VDDSD12 monitoring

VDDSD12 supply is monitored comparing it with the internal reference voltage ( $V_{REFINT}$ ). At startup, the supply voltage monitoring defaults to be active. It can then be disabled through the `SDADC12_VDD_MONITOR` option bit. Assuming  $V_{DDA}$  and  $V_{DD}$  are instantaneously available at startup, the system waits for VDDSD12 to exceed  $V_{REFINT}$  voltage before releasing the reset.

VDDSD3 is not monitored, so even if VDDSD12 is usually higher or lower than VDDSD3, the application has to make sure that VDDSD3 is set up before VDDSD12.

### Constraints on VDDSDx versus VREFSD voltage

When the reference voltage for SDADC converters ( $V_{REFSD+}$ ) is selected from SDADC power supply (`REFV[1:0]` bits of `SDADC_CR1` register set to 11), the reference is provided by converters analog supplies (VDDSD12 and VDDSD3) and VDDSD12 must be at the same voltage level as VDDSD3.

*Note:* There is one exception if SDADC1 and SDADC2 converters are disabled and SDADC3 is enabled (through `ENSDx` bits in `PWR_CR` register and `ADON` bit in `SDADC_CR2` register). In this case, VDDSD12 can be lower than VDDSD3 and the reference voltage can be provided by VDDSD3.

In STM32F378xx devices ( $V_{DD}=1.8\text{ V} \pm 8\%$ ), VDDSD12 monitoring is OFF like the other supply voltage monitoring systems. This means that the application should take care of monitoring it externally and of releasing the external power on NPOR reset pin after power-up and when all supply voltages are stable.

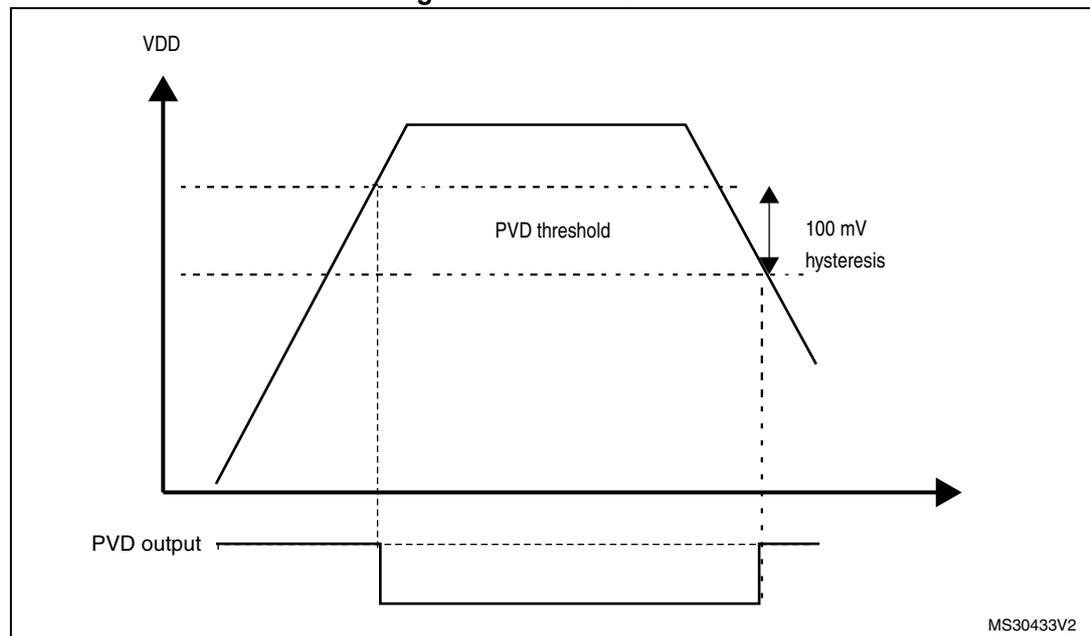
## 6.2.2 Programmable voltage detector (PVD)

You can use the PVD to monitor the  $V_{DD}$  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  $V_{DD}$  is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when  $V_{DD}$  drops below the PVD threshold and/or when  $V_{DD}$  rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

Figure 8. PVD thresholds



*Note:* In STM32F378xx devices ( $V_{DD} = 1.8 V \pm 8 \%$ ), the POR, PDR and PVD features are not available. The Power-on reset signal is applied on the NPOR pin. See details in the following section.

## 6.2.3 External NPOR signal

In STM32F378xx devices (powered from  $1.8 V \pm 8 \%$ ), the POR, PDR and PVD features are not available and the application must provide the reset signal to the external NPOR pin. The NPOR signal is active low, and must be driven to  $V_{SS}$  when the  $V_{DDA}$  is applied. Then, when  $V_{DD}$  is stable, it can be released (high impedance) and the internal pull-up will hold this input to  $V_{DDA}$ . The NPOR signal can also be controlled by using an open-drain driver circuitry.

In STM32F378xx devices, PB2 I/O is not available and is replaced by the NPOR functionality.

### 6.3 Low-power modes

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

The device features three low-power modes:

- Sleep mode (CPU clock off, all peripherals including Cortex<sup>®</sup>-M4 with FPU core peripherals like NVIC, SysTick, etc. are kept running)
- Stop mode (all clocks are stopped)
- Standby mode (1.8 V domain powered-off)

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

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

**Table 12. Low-power mode summary**

Mode name	Entry	wakeup	Effect on 1.8 V domain clocks	Effect on V <sub>DD</sub> 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) Specific communication peripherals on reception events (CEC, USART, I2C)	All 1.8V domain clocks OFF	HSI and HSE oscillators OFF	ON or in low-power mode (depends on <a href="#">Power control register (PWR_CR)</a> ) <sup>(1)</sup>
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			OFF

1. In STM32F378xx devices, Standby mode is not available. Stop mode is still available, but it is meaningless to distinguish between voltage regulator in low-power mode and voltage regulator in Run mode, because the regulator is not used and V<sub>DD</sub> is applied externally to the regulator output.

#### 6.3.1 Slowing down system clocks

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

For more details refer to [Section 7.4.2: Clock configuration register \(RCC\\_CFGR\)](#).

### 6.3.2 Peripheral clock gating

In Run mode, the HCLK and PCLK 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 [AHB peripheral clock enable register \(RCC\\_AHBENR\)](#)

### 6.3.3 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<sup>®</sup>-M4 with FPU 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 13](#) and [Table 14](#) 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<sup>®</sup>-M4 with FPU 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 13](#) and [Table 14](#) for more details on how to exit Sleep mode.

Table 13. 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 <sup>®</sup> -M4 with FPU System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to <a href="#">Table 28: List of vectors</a> If WFE was used for entry: Wakeup event: Refer to <a href="#">Section 11.2.3: Wakeup event management</a>
Wakeup latency	None

Table 14. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex <sup>®</sup> -M4 with FPU System Control register.
Mode exit	Interrupt: refer to <a href="#">Table 28: List of vectors</a> .
Wakeup latency	None

### 6.3.4 Stop mode

The Stop mode is based on the Cortex<sup>®</sup>-M4 with FPU deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode in STM32F373xx devices. In STM32F378xx, it is meaningless to distinguish between voltage regulator in low-power mode and voltage regulator in Run mode, because the regulator is not used and  $V_{DD}$  is applied externally to the regulator output..

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.

The I2C, CEC, and USART peripherals are an exception since they require a given kernel clock in Stop mode. In this case, when this specific clock request is ON, the power controller automatically forces the regulator to be ON as well, to prevent the device to operate in low-power mode since the regulator would not sustain the current required by the peripherals.

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

#### Entering Stop mode

Refer to [Table 15](#) 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 21.3: IWDG functional description](#) in [Section 21: Independent watchdog \(IWDG\)](#).
- real-time clock (RTC): this is configured by the RTCEN bit in the [RTC 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 [RTC domain control register \(RCC\\_BDCR\)](#).

The SDADC, 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.

*Note: If the application needs to disable the external oscillator (external clock) before entering Stop mode, the system clock source must be first switched to HSI and then clear the HSEON bit. Otherwise, if before entering Stop mode the HSEON bit is kept at 1, the security system (CSS) feature must be enabled to detect any external oscillator (external clock) failure and avoid a malfunction when entering Stop mode.*

### Exiting Stop mode

Refer to [Table 15](#) 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 15. Stop mode**

Stop mode	Description
Mode entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> <li>– Set SLEEPDEEP bit in Cortex<sup>®</sup>-M4 with FPU System Control register</li> <li>– Clear PDDS bit in Power Control register (PWR_CR)</li> <li>– Select the voltage regulator mode by configuring LPDS bit in PWR_CR</li> </ul> <p><i>Note: To enter Stop mode, all EXTI Line pending bits (in <a href="#">Pending register (EXTI_PR)</a>), all peripherals interrupt pending bits and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</i></p>
Mode exit	<p>If WFI was used for entry:</p> <ul style="list-style-type: none"> <li>– Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC).</li> <li>– Some specific communication peripherals (CEC, USART, I2C) interrupts, when programmed in wakeup mode (the peripheral must be programmed in wakeup mode and the corresponding interrupt vector must be enabled in the NVIC).</li> </ul> <p>Refer to <a href="#">Table 28: List of vectors</a>.</p> <p>If WFE was used for entry:</p> <ul style="list-style-type: none"> <li>– Any EXTI Line configured in event mode. Refer to <a href="#">Section 11.2.3: Wakeup event management</a></li> </ul>
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

### 6.3.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex<sup>®</sup>-M4 with FPU 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 RTC domain (see [Figure 6](#)).

**Caution:** In the STM32F378xx devices, the Standby mode is not available.

#### Entering Standby mode

Refer to [Table 16](#) 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 21.3: IWDG functional description](#) in [Section 21: Independent watchdog \(IWDG\)](#).
- real-time clock (RTC): this is configured by the RTCEN bit in the RTC 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 RTC 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 165: RTC 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 16](#) for more details on how to exit Standby mode.

**Table 16. Standby mode**

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– Set SLEEPDEEP in Cortex<sup>®</sup>-M4 with FPU System Control register</li> <li>– Set PDDS bit in Power Control register (PWR_CR)</li> <li>– Clear WUF bit in Power Control/Status register (PWR_CSR)</li> </ul>
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<sup>®</sup>-M4 with FPU 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 31.16.1: Debug support for low-power modes](#).

## 6.3.6 Auto-wakeup from low-power mode

The RTC can be used to wakeup the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop 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

*RTC domain control register (RCC\_BDCR):*

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

## 6.4 Power control registers

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

### 6.4.1 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ENSD <sub>3</sub>	ENSD <sub>2</sub>	ENSD <sub>1</sub>	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
				rw	rw	rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **ENSD3**: Enable SDADC3.

This bit is set and cleared by software.

0: SDADC3 disabled. SDADC3 is in power down mode.

1: SD3 is enabled.

Bit 10 **ENSD2**: Enable SDADC2.

This bit is set and cleared by software.

0: SDADC2 disabled. SDADC2 is in power down mode.

1: SD2 is enabled.

Bit 9 **ENSD1**: Enable SDADC1.

This bit is set and cleared by software.

0: SDADC1 disabled. SDADC1 is in power down mode.

1: SD1 is enabled.

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

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

0: Access to RTC and Backup registers disabled

1: Access to RTC and Backup registers enabled

*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.

*Note:* 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.

*Note:* Once the PVD\_LOCK is enabled (for CLASS B protection) the PLS[2:0] bits cannot be programmed anymore.

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

### 6.4.2 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	EWUP 3	EWUP 2	EWUP 1	Res.	Res.	Res.	Res.	VREFI NTRD YF	PVDO	SBF	WUF
					rw	rw	rw					r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **EWUP3**: Enable WKUP3 pin

This bit is set and cleared by software.

0: WKUP3 pin is used for general purpose I/O. An event on the WKUP3 pin does not wakeup the device from Standby mode.

1: WKUP3 pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP3 pin wakes-up the system from Standby mode).

*Note: This bit is reset by a system Reset.*

Bit 9 **EWUP2**: Enable WKUP2 pin

This bit is set and cleared by software.

0: WKUP2 pin is used for general purpose I/O. An event on the WKUP2 pin does not wakeup the device from Standby mode.

1: WKUP2 pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP2 pin wakes-up the system from Standby mode).

*Note: This bit is reset by a system Reset.*

Bit 8 **EWUP1**: Enable WKUP1 pin

This bit is set and cleared by software.

0: WKUP1 pin is used for general purpose I/O. An event on the WKUP1 pin does not wakeup the device from Standby mode.

1: WKUP1 pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP1 pin wakes-up the system from Standby mode).

*Note: This bit is reset by a system Reset.*

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **VREFINTRDY**: VREFINT reference voltage ready.

This bit is cleared and set by hardware.

This bit indicates the state of the internal reference voltage VREFINT. It is set when VREFINT is ready. It is reset during stabilization of VREFINT.

*Note: This flag is useful only for the STM32F378xx product when working with external NPOR pin. In the STM32F373xx product, the internal POR waits for the VREFINT stabilization before releasing the reset.*

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

1:  $V_{DD}$  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.*

*Once the PVD is enabled and configured in the PWR\_CR register, PVDO can be used to generate an interrupt through the Extended Interrupt/event controller.*

*Once the PVD\_LOCK is enabled (for CLASS B protection) PVDO cannot be disabled anymore.*

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 by a system 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.*

### 6.4.3 PWR register map

The following table summarizes the PWR registers.

**Table 17. 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	Res.	ENSD3	ENSD2	ENSD1	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS																				
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0
0x004	PWR_CSR	Res.	EWUP3	EWUP2	EWUP1	Res.	Res.	Res.	Res.	VREFINTRDYF	PVDO	SBF	WUF																					
	Reset value																							0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2 on page 39](#) for the register boundary addresses.

## 7 Reset and clock control (RCC)

### 7.1 Reset

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

#### 7.1.1 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 RTC domain ([Figure 6 on page 75](#)).

In STM32F378xx devices, the POR/PDR reset is not functional and the Standby mode is not available. Power reset must be provided from an external NPOR pin (active low and released by the application when all supply voltages are stabilized).

#### 7.1.2 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 RTC domain (see [Figure 6 on page 75](#)).

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

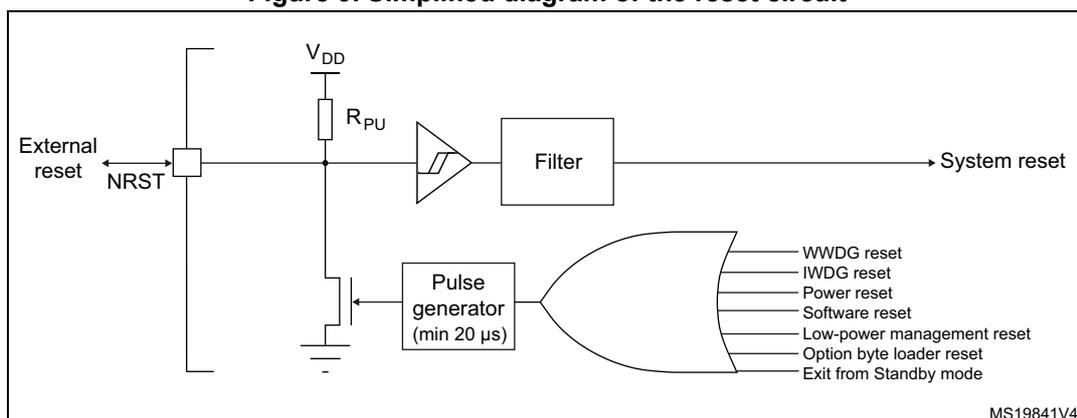
1. A low level on the NRST pin (external reset)
2. Window watchdog event (WWDG reset)
3. Independent watchdog event (IWDG reset)
4. A software reset (SW reset) (see [Software reset](#))
5. Low-power management reset (see [Low-power management reset](#))
6. Option byte loader reset (see [Option byte loader reset](#))
7. A power reset

The reset source can be identified by checking the reset flags in the Control/Status register, RCC\_CSR (see [Section 7.4.10: Control/status register \(RCC\\_CSR\)](#)).

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.

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  $\mu$ s for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Figure 9. Simplified diagram of the reset circuit



### Software reset

The SYSRESETREQ bit in Cortex<sup>®</sup>-M4 with FPU Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the *Cortex<sup>®</sup>-M0 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 [Section 4: Option bytes](#).

### Option byte loader reset

The option byte loader reset is generated when the OBL\_LAUNCH bit (bit 13) is set in the FLASH\_CR register. This bit is used to launch the option byte loading by software.

## 7.1.3 RTC domain reset

The RTC domain has two specific resets that affect only the RTC domain ([Figure 6 on page 75](#)).

An RTC domain reset only affects the LSE oscillator, the RTC, the Backup registers and the RCC [RTC domain control register \(RCC\\_BDCR\)](#). It is generated when one of the following events occurs.

1. Software reset, triggered by setting the BDRST bit in the [RTC domain control register \(RCC\\_BDCR\)](#).
2. V<sub>DD</sub> power-up if V<sub>BAT</sub> has been disconnected when it was low.

The Backup registers are also reset when one of the following events occurs:

1. RTC tamper detection event.
2. Change of the read out protection from level 1 to level 0.

## 7.2 Clocks

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

- HSI 8 MHz RC oscillator clock
- HSE oscillator clock
- PLL clock

The devices have the following additional 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.

Several prescalers allow to configure the frequency of the AHB, the high speed APB (APB2) and the low speed APB (APB1) domains. The AHB and the high speed APB domains maximum frequency is 72 MHz, while the low speed APB domain maximum frequency is 36 MHz.

The Cortex<sup>®</sup> system timer is always clocked by the AHB clock divided by 8 or directly by the AHB clock (through Cortex<sup>®</sup> Systick configuration bits).

All the peripheral clocks are derived from their bus clock (HCLK or PCLK) except:

- The Flash memory programming interface clock (FLITFCLK) which is always the HSI clock.
- The option byte loader clock which is always the HSI clock
- The ADC clock is the high speed APB clock (APB2) divided by 2, 4, 6 or 8.
- The USART1/2/3 clock which is derived (selected by software) from one of the four following sources:
  - system clock
  - HSI clock
  - LSE clock
  - APB clock (PCLK)
- The I2C1/2 clock which is derived (selected by software) from one of the two following sources:
  - system clock
  - HSI clock
- The CEC clock which is derived from the HSI clock divided by 244 or from the LSE clock.
- The I2S clock which is always the system clock.
- The RTC clock which is derived from the LSE, LSI or from the HSE clock divided by 32.
- The IWDG clock which is always the LSI clock.
- The timer clock frequencies are twice the frequency of the APB domain to which they are connected. Nevertheless, if the APB prescaler is 1, the timer clock frequency is the same as the frequency of the APB domain to which it is connected.

The RCC feeds the Cortex<sup>®</sup> System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or directly with the Cortex<sup>®</sup> clock (HCLK), configurable in the SysTick Control and Status Register.

Figure 10. Clock tree part 1

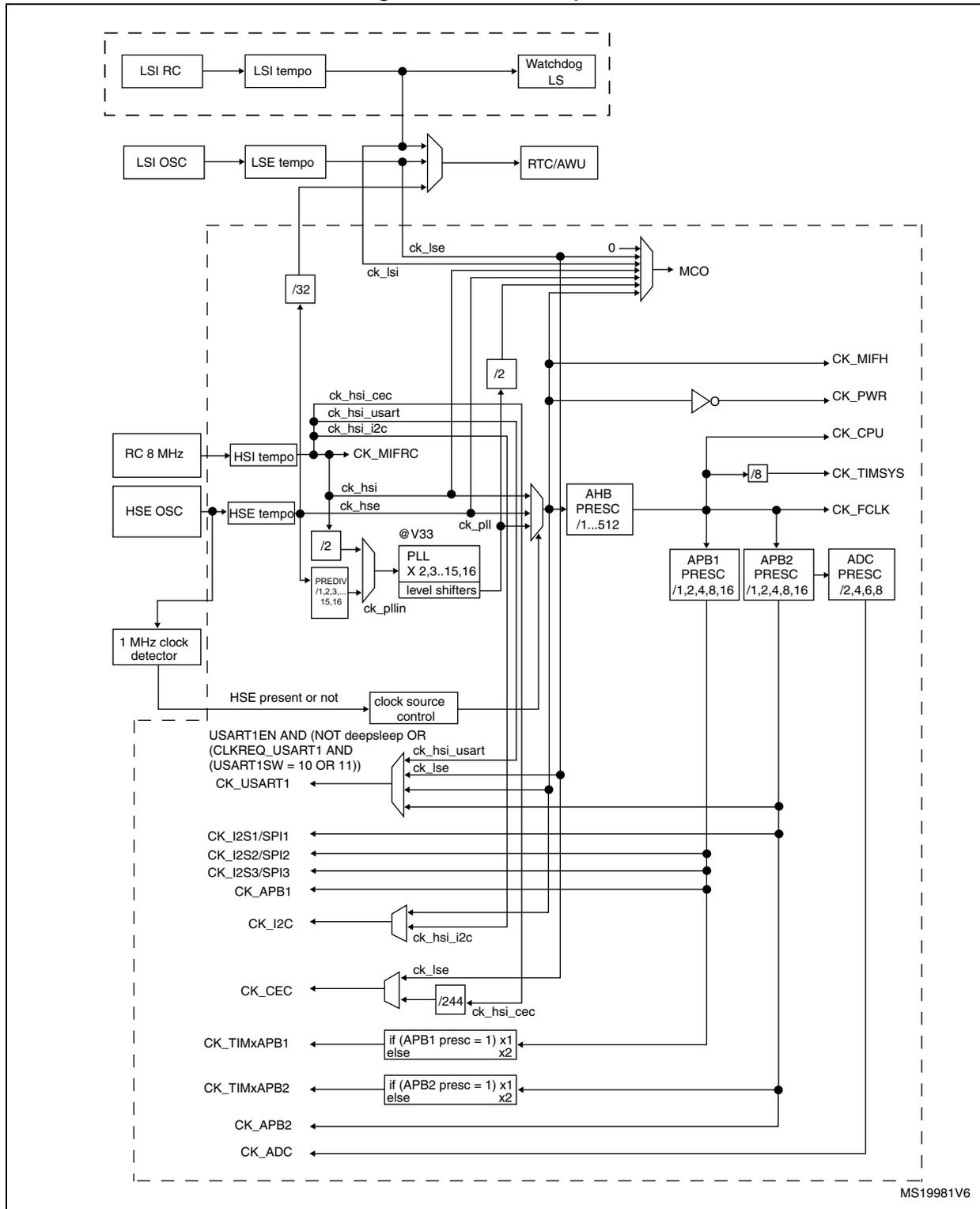
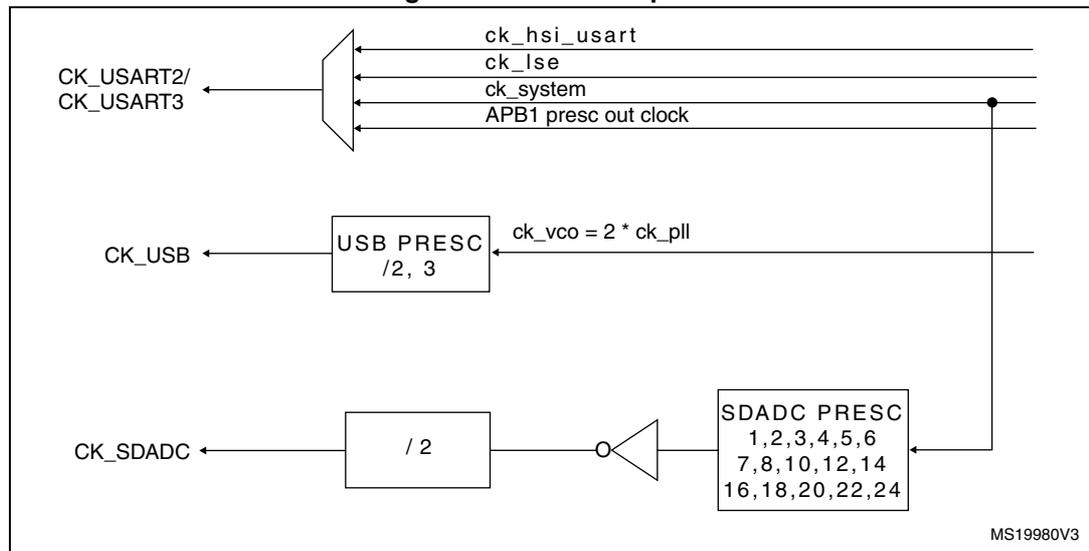


Figure 11. Clock tree part 2



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

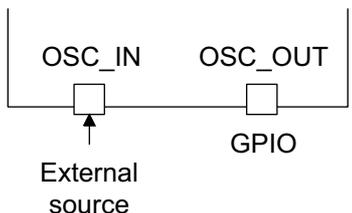
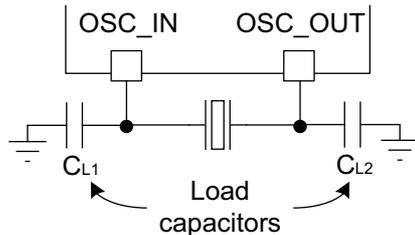
### 7.2.1 HSE clock

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 12. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	 <p style="text-align: right; font-size: small;">MSv31915V1</p>
Crystal/Ceramic resonators	 <p style="text-align: right; font-size: small;">MSv31916V1</p>

**External crystal/ceramic resonator (HSE crystal)**

The 4 to 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in [Figure 12](#). Refer to the electrical characteristics section of the *datasheet* for more details.

The HSERDY flag in the [Clock control register \(RCC\\_CR\)](#) indicates if the HSE oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the [Clock interrupt register \(RCC\\_CIR\)](#).

The HSE crystal can be switched on and off using the HSEON bit in the [Clock control register \(RCC\\_CR\)](#).

**Caution:** To switch ON the HSE oscillator, 512 HSE clock pulses need to be seen by an internal stabilization counter after the HSEON bit is set. Even in the case that no crystal or resonator is connected to the device, excessive external noise on the OSC\_IN pin may still lead the oscillator to start. Once the oscillator is started, it needs another 6 HSE clock pulses to complete a switching OFF sequence. If for any reason the oscillations are no more present on the OSC\_IN pin, the oscillator cannot be switched OFF, locking the OSC pins from any other use and introducing unwanted power consumption. To avoid such situation, it is strongly recommended to always enable the Clock Security System (CSS) which is able to switch OFF the oscillator even in this case.

**External source (HSE bypass)**

In this mode, an external clock source must be provided. It can have a frequency of up to 32 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 ~40-60% duty cycle depending on the frequency (refer to the *datasheet*) has to drive the OSC\_IN pin while the OSC\_OUT pin can be used as GPIO. See [Figure 12](#).

## 7.2.2 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  $T_A=25^{\circ}\text{C}$ .

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 \(RCC\\_CR\)](#).

For more details on how to measure the HSI frequency variation please refer to [Section 7.2.13: Internal/external clock measurement using TIM14 on page 105](#).

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 7.2.7: Clock security system \(CSS\) on page 103](#).

## 7.2.3 PLL

The internal PLL can be used to multiply the HSI or HSE output clock frequency. Refer to [Figure 10](#) and [Clock control register \(RCC\\_CR\)](#).

The PLL configuration (selection of the input clock, and multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0.
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.

An interrupt can be generated when the PLL is ready, if enabled in the [Clock interrupt register \(RCC\\_CIR\)](#).

The PLL output frequency must be set in the range 16-72 MHz.

## 7.2.4 LSE clock

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

The LSE crystal is switched on and off using the LSEON bit in *RTC domain control register (RCC\_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *RTC domain control register (RCC\_BDCR)* to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other.

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

**Caution:** To switch ON the LSE oscillator, 4096 LSE clock pulses need to be seen by an internal stabilization counter after the LSEON bit is set. Even in the case that no crystal or resonator is connected to the device, excessive external noise on the OSC32\_IN pin may still lead the oscillator to start. Once the oscillator is started, it needs another 6 LSE clock pulses to complete a switching OFF sequence. If for any reason the oscillations are no more present on the OSC\_IN pin, the oscillator cannot be switched OFF, locking the OSC32 pins from any other use and introducing unwanted power consumption. The only way to recover such situation is to perform the RTC domain reset by software.

### External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *RTC 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 can be used as GPIO. See *Figure 12*.

## 7.2.5 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent window watchdog (IWDG) and RTC. The clock frequency is around 40 kHz (between 30 kHz and 60 kHz). For more details, refer to the electrical characteristics section of the datasheets.

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

The LSIRDY flag in the *Control/status register (RCC\_CSR)* indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt register (RCC\_CIR)*.

## 7.2.6 System clock (SYSCLK) selection

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

- HSI oscillator
- HSE oscillator
- PLL

After a system reset, the HSI oscillator is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source becomes ready. Status bits in the

*Clock control register (RCC\_CR)* indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

### 7.2.7 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 TIM15/TIM16/TIM17 timers 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<sup>®</sup>-M4 with FPU 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.

### 7.2.8 ADC clock

The ADC clock is derived from the APB2 high speed clock divided by 2,4,6,8 (duty cycle 50%).

### 7.2.9 SDADC clock

The SDADC clock source is derived from the system clock divided by a selectable divider with a 50% duty cycle. Possible division factors are 2, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32, 36, 40, 44, and 48.

The SDADC clock is automatically stopped in deepsleep mode.

The maximum and minimum operating frequencies of the SDAC are 6 MHz and 500 kHz, respectively.

### 7.2.10 RTC clock

The RTCCLK clock source can be either the HSE/32, LSE or LSI clocks. This is selected by programming the RTCSEL[1:0] bits in the *RTC domain control register (RCC\_BDCR)*. This selection cannot be modified without resetting the RTC domain. The system must be always configured in a way that the PCLK frequency is greater than or equal to the RTCCLK frequency for proper operation of the RTC.

The LSE clock is in the RTC domain, whereas the HSE and LSI clocks are not. Consequently:

- If LSE is selected as RTC clock:
  - The RTC continues to work even if the  $V_{DD}$  supply is switched off, provided the  $V_{BAT}$  supply is maintained.
- If LSI is selected as the RTC clock:
  - The RTC state is not guaranteed if the  $V_{DD}$  supply is powered off.
- If the HSE clock divided by 32 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).

### 7.2.11 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.

### 7.2.12 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 5 clock signals can be selected as the MCO clock.

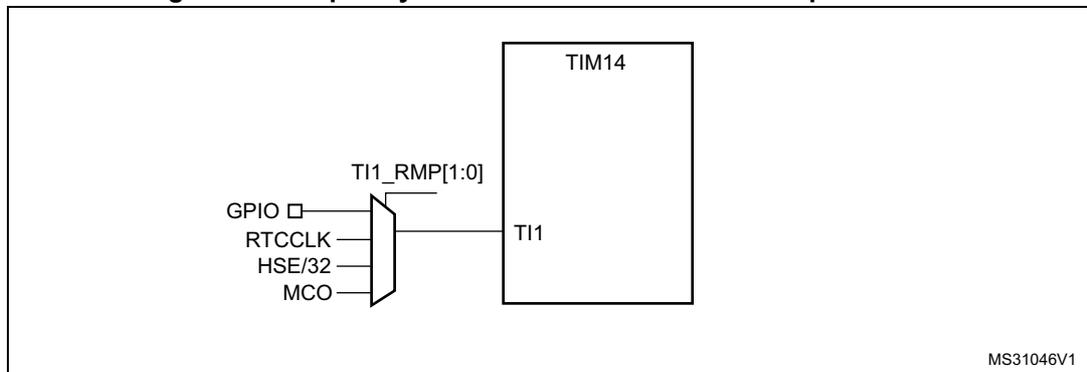
- LSI
- LSE
- SYSCLK
- HSI
- HSE
- PLL clock divided by 2

The selection is controlled by the MCO[2:0] bits of the [Clock configuration register \(RCC\\_CFGR\)](#).

### 7.2.13 Internal/external clock measurement using TIM14

It is possible to indirectly measure the frequency of all on-board clock sources by mean of the TIM14 channel 1 input capture. As represented on [Figure 13](#).

**Figure 13. Frequency measurement with TIM14 in capture mode**



The input capture channel of the Timer 14 can be a GPIO line or an internal clock of the MCU. This selection is performed through the TI1\_RMP [1:0] bits in the TIM14\_OR register. The possibilities available are the following ones.

- TIM14 Channel1 is connected to the GPIO. Refer to the alternate function mapping in the device datasheets.
- TIM14 Channel1 is connected to the RTCCLK.
- TIM14 Channel1 is connected to the HSE/32 Clock.
- TIM14 Channel1 is connected to the microcontroller clock output (MCO). Refer to section [Section 7.2.12: Clock-out capability](#) for MCO clock configuration.

#### Calibration of the HSI

The primary purpose of connecting the LSE, through the MCO multiplexer, to the channel 1 input capture is to be able to precisely measure the HSI system clocks (for this, the HSI should be used as the system clock source). The number of HSI clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm's), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process- and/or temperature- and voltage-related frequency deviations.

The HSI oscillator has dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (e.g. the HSI/LSE ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement will be.

If LSE is not available, HSE/32 will be the better option in order to reach the most precise calibration possible.

#### Calibration of the LSI

The calibration of the LSI will follow the same pattern that for the HSI, but changing the reference clock. It will be necessary to connect LSI clock to the channel 1 input capture of the TIM14. Then define the HSE as system clock source, the number of its clock counts

between consecutive edges of the LSI signal provides a measure of the internal low speed clock period.

The basic concept consists in providing a relative measurement (e.g. the HSE/LSI ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio is, the better the measurement will be.

## 7.3 Low-power modes

APB peripheral clocks and DMA clock can be disabled by software.

Sleep mode stops the CPU clock. The memory interface clocks (Flash and RAM interfaces) can be stopped by software during sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

Stop mode stops all the clocks in the V18 domain and disables the PLL and the HSI, and HSE oscillators.

HDMI CEC, USART1/2/3, and I2C1/2 have the capability to enable the HSI oscillator even when the MCU is in Stop mode (if HSI is selected as the clock source for that peripheral).

HDMI CEC and USART1/2/3 can also be driven by the LSE oscillator when the system is in Stop mode (if LSE is selected as clock source for that peripheral) and the LSE oscillator is enabled (LSEON) but they do not have the capability to turn on the LSE oscillator.

Standby mode stops all the clocks in the V18 domain and disables the PLL and the HSI, and HSE oscillators.

The CPU's deepsleep mode can be overridden for debugging by setting the DBG\_STOP or DBG\_STANDBY bits in the DBGMCU\_CR register.

When waking up from deepsleep after an interrupt (Stop mode) or reset (Standby mode), the HSI oscillator is selected as system clock.

If a Flash programming operation is on going, deepsleep mode entry is delayed until the Flash interface access is finished. If an access to the APB domain is ongoing, deepsleep mode entry is delayed until the APB access is finished.

## 7.4 RCC registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

### 7.4.1 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
Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLLON	Res.	Res.	Res.	Res.	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, must be kept at reset value.

Bit 25 **PLL RDY**: 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, must be kept at reset value.

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 a 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**: HSE crystal oscillator bypass

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

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

- Bit 17 **HSERDY**: HSE 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 fall down after HSEON reset.  
0: HSE oscillator not ready  
1: HSE oscillator ready
- Bit 16 **HSEON**: HSE clock enable  
Set and cleared by software.  
Cleared by hardware to stop the HSE oscillator when entering 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]**: HSI clock calibration  
These bits are initialized automatically at startup.
- Bits 7:3 **HSITRIM[4:0]**: HSI 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 HSI.  
The default value is 16, which, when added to the HSICAL value, should trim the HSI to 8 MHz  $\pm$  1%. The trimming step ( $F_{\text{hsitrim}}$ ) is around 40 kHz between two consecutive HSICAL steps.
- Bit 2 Reserved, must be kept at reset value.
- Bit 1 **HSIRDY**: HSI clock ready flag  
Set by hardware to indicate that HSI oscillator is stable. After the HSION bit is cleared, HSIRDY goes low after 6 HSI oscillator clock cycles.  
0: HSI oscillator not ready  
1: HSI oscillator ready
- Bit 0 **HSION**: HSI clock enable  
Set and cleared by software.  
Set by hardware to force the HSI oscillator ON when leaving Stop or Standby mode or in case of failure of the HSE crystal oscillator used directly or indirectly as system clock. This bit cannot be reset if the HSI is used directly or indirectly as system clock or is selected to become the system clock.  
0: HSI oscillator OFF  
1: HSI oscillator ON

### 7.4.2 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
SDPRE[4:0]					MCO[2:0]			Res.	USBP RE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
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
ADCPRE[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 **SDPRE[4:0]**: SDADC prescaler

These bits are set and reset by software to control AHB clocks division factor.

0xxxx: system clock divided by 2  
10000: system clock divided by 2  
10001: system clock divided by 4  
10010: system clock divided by 6  
10011: system clock divided by 8  
10100: system clock divided by 10  
10101: system clock divided by 12  
10110: system clock divided by 14  
10111: system clock divided by 16  
11000: system clock divided by 20  
11001: system clock divided by 24  
11010: system clock divided by 28  
11011: system clock divided by 32  
11100: system clock divided by 36  
11101: system clock divided by 40  
11110: system clock divided by 44  
11111: system clock divided by 48

Bits 26:24 **MCO[2:0]**: Microcontroller clock output

Set and cleared by software.

000: MCO output disabled, no clock on MCO  
001: Reserved  
010: LSI clock selected  
011: LSE clock selected  
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 Reserved, must be kept at reset value.

Bit 22 **USBPRE**: USB prescaler

This bit is set and reset by software to generate the 48 MHz USB clock. They must be valid before enabling USB clocks.

0: PLL clock is divided by 1,5  
1: PLL clock is not divided

- Bits 21:18 **PLLMUL[3:0]**: 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 not exceed 72 MHz.  
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**: HSE divider for PLL input clock  
This bits is set and cleared by software to select the HSE division factor for the PLL. It can be written only when the PLL is disabled.  
0000: HSE input to PLL not divided  
0001: HSE input to PLL divided by 2  
*Note: This bit is the same as the LSB of PREDIV in [Clock configuration register 2 \(RCC\\_CFGR2\)](#) (for compatibility with other STM32 products)*
- 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/2 selected as PLL input clock  
1: HSE/PREDIV selected as PLL input clock (refer to [Section 7.4.12: Clock configuration register 2 \(RCC\\_CFGR2\)](#) on page 132)
- Bits 15:14 **ADCPRE[1:0]**: ADC prescaler  
These bits are set and cleared by software to select the frequency of the clock to the ADC.  
00: PCLK divided by 2  
01: PCLK divided by 4  
10: PCLK divided by 6  
11: PCLK divided by 8
- Bits 13:11 **PPRE2[2:0]**: APB high speed prescaler (APB2)  
These bits are set and reset by software to control the high speed APB clock division factor.  
0xx: AHB clock not divided  
100: AHB clock divided by 2  
101: AHB clock divided by 4  
110: AHB clock divided by 8  
111: AHB clock divided by 16

Bits 10:8 **PPRE1[2:0]**: APB low speed prescaler (APB1)

These bits are set and cleared by software to control the low speed APB clock division factor.

0xx: APB1 clock not divided

100: APB1 divided by 2

101: APB1 divided by 4

110: APB1 divided by 8

111: APB1 divided by 16

Bits 7:4 **HPRE[3:0]**: 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

*Note: The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock. Refer to section [Read operations on page 47](#) for more details.*

Bits 3:2 **SWS[1:0]**: System clock switch status

Set and cleared by hardware to indicate which clock source is used as system clock.

00: 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[1:0]**: System clock switch

Set and cleared by software to select SYSCLK source.

Cleared 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

### 7.4.3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	Res.	Res.	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
Res.	Res.	Res.	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	Res.	Res.	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, must be kept at reset value.

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

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **PLL RDYC**: PLL ready interrupt clear  
 This bit is set by software to clear the PLLRDYF flag.  
 0: No effect  
 1: Clear PLLRDYF flag

Bit 19 **HSE RDYC**: HSE ready interrupt clear  
 This bit is set by software to clear the HSERDYF flag.  
 0: No effect  
 1: Clear HSERDYF flag

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

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

Bit 16 **LSI RDYC**: 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, must be kept at reset value.

- 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 HSI 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 LSE 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 the LSI 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
- Bit 6:5 Reserved, must be kept at reset value.
- 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
- Bit 3 **HSERDYF**: HSE ready interrupt flag  
Set by hardware when the HSE clock becomes stable and HSERDYDIE is set.  
Cleared by software setting the HSERDYC bit.  
0: No clock ready interrupt caused by the HSE oscillator  
1: Clock ready interrupt caused by the HSE oscillator

Bit 2 **HSIRDYF**: HSI ready interrupt flag

Set by hardware when the HSI clock becomes stable and HSIRDYDIE is set in a response to setting the HSION (refer to [Clock control register \(RCC\\_CR\)](#)).  
 When HSION is not set but the HSI oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated.  
 Cleared by software setting the HSIRDYC bit.  
 0: No clock ready interrupt caused by the HSI oscillator  
 1: Clock ready interrupt caused by the HSI oscillator

Bit 1 **LSERDYF**: LSE ready interrupt flag

Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.  
 Cleared by software setting the LSERDYC bit.  
 0: No clock ready interrupt caused by the LSE oscillator  
 1: Clock ready interrupt caused by the LSE oscillator

Bit 0 **LSIRDYF**: LSI ready interrupt flag

Set by hardware when the LSI clock becomes stable and LSIRDYDIE is set.  
 Cleared by software setting the LSIRDYC bit.  
 0: No clock ready interrupt caused by the LSI oscillator  
 1: Clock ready interrupt caused by the LSI oscillator

**7.4.4 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
Res.	Res.	Res.	Res.	Res.	SDAD3 RST	SDAD2 RST	SDAD1 RST	Res.	Res.	Res.	Res.	TIM19 RST	TIM17 RST	TIM16 RST	TIM15 RST
					rw	rw	rw					rw	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	Res.	Res.	ADC RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS CFG RST
	rw		rw			rw									rw

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

Bit 26 **SDAD3RST**: SDADC3 (Sigma delta ADC 3) reset

This bit is set and reset by software.  
 0: does not reset the SDADC3  
 1: resets the SDADC3

Bit 25 **SDAD2RST**: SDADC2 (Sigma delta ADC 2) reset

This bit is set and reset by software.  
 0: does not reset the SDADC2  
 1: resets the SDADC2

- Bit 24 **SDAD1RST**: SDADC1 (Sigma delta ADC 1) reset  
This bit is set and reset by software.  
0: does not reset the SDADC1  
1: resets the SDADC1
- Bits 22:20 Reserved, must be kept at reset value.
- Bit 19 **TIM19RST**: TIM19 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM19 timer
- Bit 18 **TIM17RST**: TIM17 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM17 timer
- Bit 17 **TIM16RST**: TIM16 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM16 timer
- Bit 16 **TIM15RST**: TIM15 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM15 timer
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **USART1RST**: USART1 reset  
Set and cleared by software.  
0: No effect  
1: Reset USART1
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **SPI1RST**: SPI1 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI1
- Bit 11:10 Reserved, must be kept at reset value.
- Bit 9 **ADCRST**: ADC interface reset  
Set and cleared by software.  
0: No effect  
1: Reset ADC interface
- Bits 8:1 Reserved, must be kept at reset value.
- Bit 0 **SYSCFGRST**: SYSCFG reset  
Set and cleared by software.  
0: No effect  
1: Reset all SYSCFG registers except for SYSCFG\_CFGR2

### 7.4.5 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	DAC1RST	PWR RST	Res.	DAC2RST	CAN RST	Res.	USB RST	I2C2 RST	I2C1 RST	Res.	Res.	USART3 RST	USART2 RST	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
SPI3 RST	SPI2 RST	Res.	Res.	WWD GRST	Res.	TIM18 RST	TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

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

Bit 29 **DAC1RST**: DAC1 interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset DAC1 interface

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

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DAC2RST**: DAC2 interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset DAC2 interface

Bit 25 **CAN RST**: CAN interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset CAN interface

Bit 24 Reserved, must be kept at reset value.

Bit 23 **USB RST**: USB interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset USB interface

- Bit 22 **I2C2RST**: I2C2 reset  
Set and cleared by software.  
0: No effect  
1: Reset I2C2
- Bit 21 **I2C1RST**: I2C1 reset  
Set and cleared by software.  
0: No effect  
1: Reset I2C1
- Bits 20:19 Reserved, must be kept at reset value.
- Bit 18 **USART3RST**: USART3 reset  
Set and cleared by software.  
0: No effect  
1: Reset USART3
- Bit 17 **USART2RST**: USART2 reset  
Set and cleared by software.  
0: No effect  
1: Reset USART2
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3RST**: SPI3 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI3
- Bit 14 **SPI2RST**: SPI2 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI2
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGRST**: Window watchdog reset  
Set and cleared by software.  
0: No effect  
1: Reset window watchdog
- Bit 10 Reserved, must be kept at reset value.
- Bit 9 **TIM18RST**: TIM18 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM18
- Bit 8 **TIM14RST**: TIM14 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM14
- Bit 7 **TIM13RST**: TIM13 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM13

- Bit 6 **TIM12RST**: TIM12 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM12
- Bit 5 **TIM7RST**: TIM7 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM7
- Bit 4 **TIM6RST**: TIM6 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM6
- Bit 3 **TIM5RST**: TIM5 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM5
- Bit 2 **TIM4RST**: TIM4 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM4
- Bit 1 **TIM3RST**: TIM3 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM3
- Bit 0 **TIM2RST**: TIM2 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM2

### 7.4.6 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
Res.	TSCEN	Res.	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.						
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CRCEN	Res.	FLITFEN	Res.	SRAMEN	DMA2EN	DMAEN							
									rw		rw		rw	rw	rw



- Bits 31:25 Reserved, must be kept at reset value.
- Bit 24 **TSCEN**: Touch sensing controller clock enable  
Set and cleared by software.  
0: TSC clock disabled  
1: TSC clock enabled
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 **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 21 **IOPHEN**: 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 20 **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 19 **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 18 **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 17 **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
- Bits 16:7 Reserved, must be kept at reset value.
- Bit 6 **CRCEN**: CRC clock enable  
Set and cleared by software.  
0: CRC clock disabled  
1: CRC clock enabled
- Bit 5 Reserved, must be kept at reset value.
- 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, must be kept at reset value.

- 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 **DMAEN**: DMA clock enable  
 Set and cleared by software.  
 0: DMA clock disabled  
 1: DMA clock enabled

### 7.4.7 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 APB domain is on going. In this case, wait states are inserted until the access to APB 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.	Res.	Res.	Res.	Res.	SDAD C3EN	SDAD C2EN	SDAD C1EN	Res.	DBGM CUEN	Res.	Res.	TIM19 EN	TIM17 EN	TIM16 EN	TIM15 EN
					rw	rw	rw		rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USAR T1EN	Res.	SPI1 EN	Res.	Res.	ADC EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYS CFG EN
	rw		rw			rw									rw

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

- Bit 26 **SDADC3**: SDADC3 (Sigma Delta ADC 3) clock enable  
 Set and reset by software.  
 0: SDADC3 clock disabled  
 1: SDADC3 clock enabled
- Bit 25 **SDADC2**: SDADC2 (Sigma Delta ADC 2) clock enable  
 Set and reset by software.  
 0: SDADC2 clock disabled  
 1: SDADC2 clock enabled

- Bit 24 **SDADC1**: SDADC1 (Sigma Delta ADC 1) clock enable  
Set and reset by software.  
0: SDADC1 clock disabled  
1: SDADC1 clock enabled
- Bit 23:22 Reserved, must be kept at reset value.
- Bits 21:20 Reserved, must be kept at reset value.
- Bit 19 **TIM19EN**: TIM19 timer clock enable  
Set and cleared by software.  
0: TIM19 timer clock disabled  
1: TIM19 timer clock enabled
- 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, must be kept at reset value.
- Bit 14 **USART1EN**: USART1 clock enable  
Set and cleared by software.  
0: USART1 clock disabled  
1: USART1 clock enabled
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **SPI1EN**: SPI1 clock enable  
Set and cleared by software.  
0: SPI1 clock disabled  
1: SPI1 clock enabled
- Bit 11 Reserved, must be kept at reset value.
- Bit 10 Reserved, must be kept at reset value.
- Bit 9 **ADCEN**: ADC interface clock enable  
Set and cleared by software.  
0: ADC interface disabled  
1: ADC interface clock enabled
- Bits 8:1 Reserved, must be kept at reset value.
- Bit 0 **SYSCFGEN**: SYSCFG clock enable  
Set and cleared by software.  
0: SYSCFG clock disabled  
1: SYSCFG clock enabled

### 7.4.8 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.	CEC EN	DAC1 EN	PWR EN	Res.	DAC2 EN	CAN EN	Res.	USB EN	I2C2 EN	I2C1 EN	Res.	Res.	USART3 EN	USART2 EN	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
SPI3 EN	SPI2 EN	Res.	Res.	WWD GEN	Res.	TIM18 EN	TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Bit 31 Reserved, must be kept at reset value.
- Bit 30 **CECEN**: HDMI CEC interface clock enable  
Set and cleared by software.  
0: HDMI CEC clock disabled  
1: HDMI CEC clock enabled
- Bit 29 **DAC1EN**: DAC1 interface clock enable  
Set and cleared by software.  
0: DAC1 interface clock disabled  
1: DAC1 interface clock enabled
- Bit 28 **PWREN**: Power interface clock enable  
Set and cleared by software.  
0: Power interface clock disabled  
1: Power interface clock enabled
- Bit 27 Reserved, must be kept at reset value.
- Bit 26 **DAC2EN**: DAC2 interface clock enable  
Set and cleared by software.  
0: DAC2 interface clock disabled  
1: DAC2 interface clock enabled
- Bit 25 **CANEN**: CAN interface clock enable  
Set and cleared by software.  
0: CAN interface clock disabled  
1: CAN interface clock enabled
- Bit 24 Reserved, must be kept at reset value.

- Bit 23 **USBEN**: USB interface clock enable  
Set and cleared by software.  
0: USB interface clock disabled  
1: USB interface clock enabled
- Bit 22 **I2C2EN**: I2C2 clock enable  
Set and cleared by software.  
0: I2C2 clock disabled  
1: I2C2 clock enabled
- Bit 21 **I2C1EN**: I2C1 clock enable  
Set and cleared by software.  
0: I2C1 clock disabled  
1: I2C1 clock enabled
- Bits 20:18 Reserved, must be kept at reset value.
- Bit 17 **USART2EN**: USART2 clock enable  
Set and cleared by software.  
0: USART2 clock disabled  
1: USART2 clock enabled
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **SPI3EN**: SPI3 clock enable  
Set and cleared by software.  
0: SPI3 clock disabled  
1: SPI3 clock enabled
- Bit 14 **SPI2EN**: SPI2 clock enable  
Set and cleared by software.  
0: SPI2 clock disabled  
1: SPI2 clock enabled
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WWDGEN**: Window watchdog clock enable  
Set and cleared by software.  
0: Window watchdog clock disabled  
1: Window watchdog clock enabled
- Bit 10 Reserved, must be kept at reset value.
- Bit 9 **TIM18EN**: TIM18 timer clock enable  
Set and cleared by software.  
0: TIM18 clock disabled  
1: TIM18 clock enabled
- Bit 8 **TIM14EN**: TIM14 timer clock enable  
Set and cleared by software.  
0: TIM14 clock disabled  
1: TIM14 clock enabled
- Bit 7 **TIM13EN**: TIM13 timer clock enable  
Set and cleared by software.  
0: TIM13 clock disabled  
1: TIM13 clock enabled

- Bit 6 **TIM12EN**: TIM12 timer clock enable  
Set and cleared by software.  
0: TIM12 clock disabled  
1: TIM12 clock enabled
- Bit 5 **TIM7EN**: TIM7 timer clock enable  
Set and cleared by software.  
0: TIM7 clock disabled  
1: TIM7 clock enabled
- Bit 4 **TIM6EN**: TIM6 timer clock enable  
Set and cleared by software.  
0: TIM6 clock disabled  
1: TIM6 clock enabled
- Bit 3 **TIM5EN**: TIM5 timer clock enable  
Set and cleared by software.  
0: TIM5 clock disabled  
1: TIM5 clock enabled
- Bit 2 **TIM4EN**: TIM4 timer clock enable  
Set and cleared by software.  
0: TIM4 clock disabled  
1: TIM4 clock enabled
- Bit 1 **TIM3EN**: TIM3 timer clock enable  
Set and cleared by software.  
0: TIM3 clock disabled  
1: TIM3 clock enabled
- Bit 0 **TIM2EN**: TIM2 timer clock enable  
Set and cleared by software.  
0: TIM2 clock disabled  
1: TIM2 clock enabled

### 7.4.9 RTC domain control register (RCC\_BDCR)

Address offset: 0x20

Reset value: 0x0000 0000, reset by RTC 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:* The LSEON, LSEBYP, RTCSEL and RTCEN bits of the [RTC domain control register \(RCC\\_BDCR\)](#) are in the RTC 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 6.1.2 on page 78](#) for further information. These bits are only reset after a RTC domain Reset (see [Section 7.1.3: RTC 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	Res.	Res.	LSEDRV[1:0]		LSE BYP	LSE RDY	LSEON
rw						rw	rw				rw	rw	rw	r	rw

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

Bit 16 **BDRST**: RTC domain software reset

Set and cleared by software.

0: Reset not activated

1: Resets the entire RTC domain

Bit 15 **RTCEN**: RTC clock enable

Set and cleared by software.

0: RTC clock disabled

1: RTC clock enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**: RTC clock source selection

Set by software to select the clock source for the RTC. Once the RTC clock source has been selected, it cannot be changed anymore unless the RTC 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 32 used as RTC clock

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

- Bits 3:3 **LSEDRV** LSE oscillator drive capability  
Set and reset by software to modulate the LSE oscillator's drive capability. A reset of the RTC domain restores the default value.  
00: 'Xtal mode' lower driving capability  
01: 'Xtal mode' medium low driving capability  
10: 'Xtal mode' medium high driving capability  
11: 'Xtal mode' higher driving capability (reset value)  
*Note: The oscillator is in Xtal mode when it is not in bypass mode.*
- Bit 2 **LSEBYP**: LSE oscillator bypass  
Set and cleared by software to bypass oscillator 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**: LSE oscillator ready  
Set and cleared by hardware to indicate when the external 32 kHz oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.  
0: LSE oscillator not ready  
1: LSE oscillator ready
- Bit 0 **LSEON**: LSE oscillator enable  
Set and cleared by software.  
0: LSE oscillator OFF  
1: LSE oscillator ON

### 7.4.10 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	OB LRSTF	RMVF	Res.	Res.						
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 window watchdog reset flag

Set by hardware when an independent watchdog reset from V<sub>DD</sub> 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 **OBLRSTF**: Option byte loader reset flag  
Set by hardware when a reset from the OBL occurs.  
Cleared by writing to the RMVF bit.  
0: No reset from OBL occurred  
1: Reset from OBL occurred
- 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, must be kept at reset value.
- Bit 1 **LSIRDY**: LSI oscillator ready  
Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles.  
0: LSI oscillator not ready  
1: LSI oscillator ready
- Bit 0 **LSION**: LSI oscillator enable  
Set and cleared by software.  
0: LSI oscillator OFF  
1: LSI oscillator ON

**7.4.11 AHB peripheral reset register (RCC\_AHBRSTR)**

Address: 0x28

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSC RST	Res.	IOPF RST	IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.						
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCRST**: Touch sensing controller reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset TSC

Bit 23 Reserved, must be kept at reset value.

Bit 22 **IOPFRST**: I/O port F reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset I/O port F

Bit 21 **IOPERST**: I/O port E reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset I/O port E

Bit 20 **IOPDRST**: I/O port D reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset I/O port D

Bit 19 **IOPCRST**: I/O port C reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset I/O port C

Bit 18 **IOPBRST**: I/O port B reset  
Set and cleared by software.  
0: No effect  
1: Reset I/O port B

Bit 17 **IOPARST**: I/O port A reset  
Set and cleared by software.  
0: No effect  
1: Reset I/O port A

Bits 16:0 Reserved, must be kept at reset value.

**7.4.12 Clock configuration register 2 (RCC\_CFGR2)**

Address: 0x2C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	PREDIV[3:0]															
													r/w	r/w	r/w	r/w

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

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

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

*Note: Bit 0 is the same bit as bit17 in Clock configuration register (RCC\_CFGR), so modifying bit17 Clock configuration register (RCC\_CFGR) also modifies bit 0 in Clock configuration register 2 (RCC\_CFGR2) (for compatibility with other STM32 products)*

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

### 7.4.13 Clock configuration register 3 (RCC\_CFGR3)

Address: 0x30

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	USART3SW[1:0]		USART2SW[1:0]										
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEC SW	I2C2 SW	I2C1 SW	Res.	Res.	USART1SW[1:0]									
									rw	rw	rw			rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:18 **USART3SW[1:0]**: USART3 clock source selection

This bit is set and cleared by software to select the USART3 clock source.

00: PCLK selected as USART3 clock source (default)

01: System clock (SYSCLK) selected as USART3 clock

10: LSE clock selected as USART3 clock

11: HSI clock selected as USART3 clock

Bits 17:16 **USART2SW[1:0]**: USART2 clock source selection

This bit is set and cleared by software to select the USART2 clock source.

00: PCLK selected as USART2 clock source (default)

01: System clock (SYSCLK) selected as USART2 clock

10: LSE clock selected as USART2 clock

11: HSI clock selected as USART2 clock

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **CECSW**: HDMI CEC clock source selection

This bit is set and cleared by software to select the CEC clock source.

0: HSI clock, divided by 244, selected as CEC clock (default)

1: LSE clock selected as CEC clock

Bit 5 **I2C2SW**: I2C2 clock source selection

This bit is set and cleared by software to select the I2C2 clock source.

0: HSI clock selected as I2C2 clock source (default)

1: System clock (SYSCLK) selected as I2C2 clock

Bit 4 **I2C1SW**: I2C1 clock source selection

This bit is set and cleared by software to select the I2C1 clock source.

0: HSI clock selected as I2C1 clock source (default)

1: System clock (SYSCLK) selected as I2C1 clock

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **USART1SW[1:0]**: USART1 clock source selection

This bit is set and cleared by software to select the USART1 clock source.

00: PCLK selected as USART1 clock source (default)

01: System clock (SYSCLK) selected as USART1 clock

10: LSE clock selected as USART1 clock

11: HSI clock selected as USART1 clock

### 7.4.14 RCC register map

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

**Table 18. 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	
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLL ON	Res.	Res.	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]							HSITRIM[4:0]				Res.	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	
0x04	RCC_CFGR	SDPRE[4:0]					MCO [2:0]			Res.	USBPRE	PLLMUL[3:0]			Res.	PLLXTPRE	PLLSRC	ADC PRE [1:0]	PPRE2 [2:0]			PPRE [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	0	0	0		
0x08	RCC_CIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSC	Res.	Res.	Res.	PLL RDYC	HSE RDYC	HSIRDYC	LSERDYC	LSIRDYC	Res.	Res.	Res.	PLL RDYIE	HSE RDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Res.	Res.	Res.	Res.	Res.		
	Reset value									0				0	0	0	0	0				0	0	0	0	0	0							
0x0C	RCC_APB2RS TR	Res.	Res.	Res.	Res.	Res.	SDAD3RST	SDAD2RST	SDAD1RST	Res.	Res.	Res.	Res.	TIM19RST	TIM17RST	TIM16RST	TIM15RST	Res.	USART1RST	Res.	Res.	SP11RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value						0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	RCC_APB1RS TR	Res.	CECRST	DAC1RST	PWRRST	Res.	DAC2RST	CANRST	Res.	USBRST	I2C2RST	I2C1RST	Res.	Res.	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0		0	0		0	0	0			0	0		0	0															
0x14	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCEN	Res.	IOPFEN	IOPEEN	IOPDEN	IOPCEN	IOPBEN	IOPAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value								0		0	0	0	0	0	0											0							
0x18	RCC_APB2EN R	Res.	Res.	Res.	Res.	Res.	SDADC3EN	SDADC2EN	SDADC1EN	Res.	Res.	Res.	Res.	TIM19 EN	TIM17 EN	TIM16 EN	TIM15 EN	Res.	USART1EN	Res.	Res.	SPI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value						0	0	0					0	0	0	0		0			0												
0x1C	RCC_APB1EN R	Res.	CECEN	DAC1EN	PWREN	Res.	DAC2EN	CANEN	Res.	USBEN	I2C2EN	I2C1EN	Res.	Res.	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value		0	0	0		0	0		0	0	0			0	0		0	0															
0x20	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																0	0																



Table 18. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x24	RCC_CSR	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	OBLRSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value	0	0	0	0	1	1	0	0																							0	0	
0x28	RCC_AHBRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC RST	Res.	IOPFRST	IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.																	
	Reset value							0		0	0	0	0	0	0	0																		
0x2C	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV[3:0]	
	Reset value																																0	
0x30	RCC_CFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1SW[1:0]
	Reset value														0	0	0	0															0	0

Refer to [Section 2.2.2 on page 39](#) for the register boundary addresses.



## 8 General-purpose I/Os (GPIO)

### 8.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR and GPIOx\_PUPDR), two 32-bit data registers (GPIOx\_IDR and GPIOx\_ODR), a 32-bit set/reset register (GPIOx\_BSRR), a 32-bit locking register (GPIOx\_LCKR) and two 32-bit alternate function selection registers (GPIOx\_AFRH and GPIOx\_AFRL).

### 8.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx\_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx\_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx\_BSRR) for bitwise write access to GPIOx\_ODR
- Locking mechanism (GPIOx\_LCKR) provided to freeze the port A, B and D I/O port configuration .
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every one AHB clock cycle
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

### 8.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

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

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx\_BSRR and GPIOx\_BRR registers is to allow atomic read/modify accesses to any of the GPIOx\_ODR

registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 14 and Figure 15 show the basic structures of a standard and a 5 V tolerant I/O port bit, respectively. Table 20 gives the possible port bit configurations.

Figure 14. Basic structure of an I/O port bit

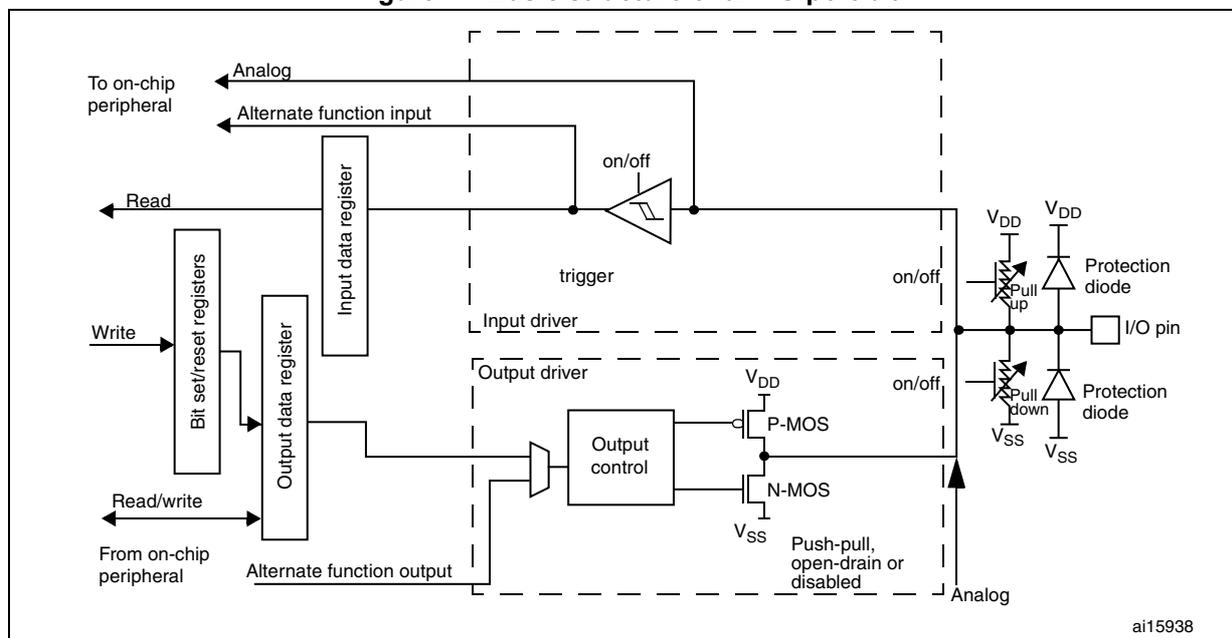
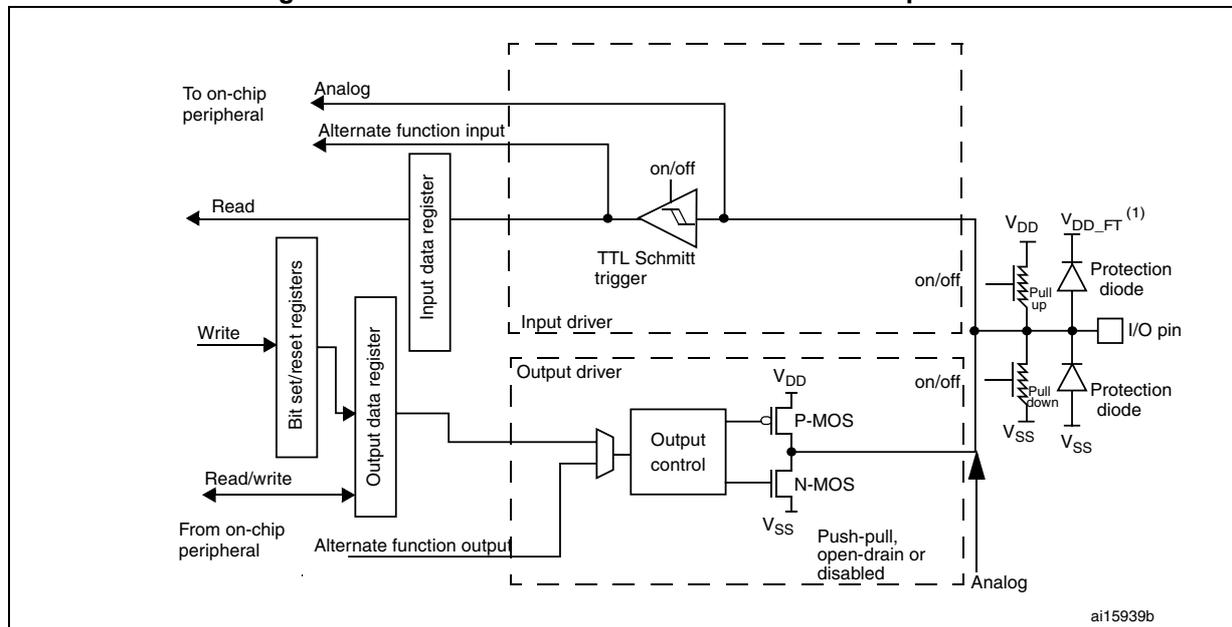


Figure 15. Basic structure of a five-volt tolerant I/O port bit



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

Table 19. Port bit configuration table<sup>(1)</sup>

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [1:0]		PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	Reserved (GP output OD)		
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	Reserved		
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

### 8.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in input floating mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK in pull-down
- PA13: JTMS in pull-up
- PB4: NJTRST in pull-up

When the pin is configured as output, the value written to the output data register (GPIOx\_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx\_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx\_PUPDR register.

### 8.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) that can be configured through the GPIOx\_AFRL (for pin 0 to 7) and GPIOx\_AFRH (for pin 8 to 15) registers:

- After reset all I/Os are connected to alternate function 0 (AF0)
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, you have to proceed as follows:

- **Debug function:** after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx\_MODER register.
- **Peripheral alternate function:**
  - Connect the I/O to the desired AFx in one of the GPIOx\_AFRL or GPIOx\_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx\_OTYPER, GPIOx\_PUPDR and GPIOx\_OSPEEDER registers, respectively.
  - Configure the desired I/O as an alternate function in the GPIOx\_MODER register.
- **Additional functions:**
  - For the ADC and DACs, configure the desired I/O in analog mode in the GPIOx\_MODER register and configure the required function in the ADC or DAC registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.
- **EVENTOUT**

You can configure the I/O pin used to output the Cortex<sup>®</sup>-M4 with FPU EVENTOUT signal by connecting it to AF15.

*Note:* *EVENTOUT is not mapped onto the following I/O pins: PC13, PC14, PC15, PF0, and PF1.*

Please refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

### 8.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR) to configure up to 16 I/Os. The GPIOx\_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx\_OTYPER and GPIOx\_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx\_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

### 8.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx\_IDR and GPIOx\_ODR). GPIOx\_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx\_IDR), a read-only register.

See [Section 8.4.5: GPIO port input data register \(GPIOx\\_IDR\) \(x = A..F\)](#) and [Section 8.4.6: GPIO port output data register \(GPIOx\\_ODR\) \(x = A..F\)](#) for the register descriptions.

### 8.3.5 I/O data bitwise handling

The bit set reset register (GPIOx\_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx\_ODR). The bit set reset register has twice the size of GPIOx\_ODR.

To each bit in GPIOx\_ODR, correspond two control bits in GPIOx\_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx\_BSRR does not have any effect on the corresponding bit in GPIOx\_ODR. If there is an attempt to both set and reset a bit in GPIOx\_BSRR, the set action takes priority.

Using the GPIOx\_BSRR register to change the values of individual bits in GPIOx\_ODR is a “one-shot” effect that does not lock the GPIOx\_ODR bits. The GPIOx\_ODR bits can always be accessed directly. The GPIOx\_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx\_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

### 8.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx\_LCKR register. The frozen registers are GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR, GPIOx\_AFRL and GPIOx\_AFRH.

To write the GPIOx\_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next reset. Each GPIOx\_LCKR bit freezes the corresponding bit in the control registers (GPIOx\_MODER, GPIOx\_OTYPER, GPIOx\_OSPEEDR, GPIOx\_PUPDR, GPIOx\_AFRL and GPIOx\_AFRH).

The LOCK sequence (refer to [Section 8.4.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A, B, and D\)](#)) can only be performed using a word (32-bit long) access to the GPIOx\_LCKR register due to the fact that GPIOx\_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details please refer to LCKR register description in [Section 8.4.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A, B, and D\)](#).

### 8.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, you can connect an alternate function to some other pin as required by your application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx\_AFRL and GPIOx\_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

### 8.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode. Refer to [Section 11.2: Extended interrupts and events controller \(EXTI\)](#) and to [Section 11.2.3: Wakeup event management](#).

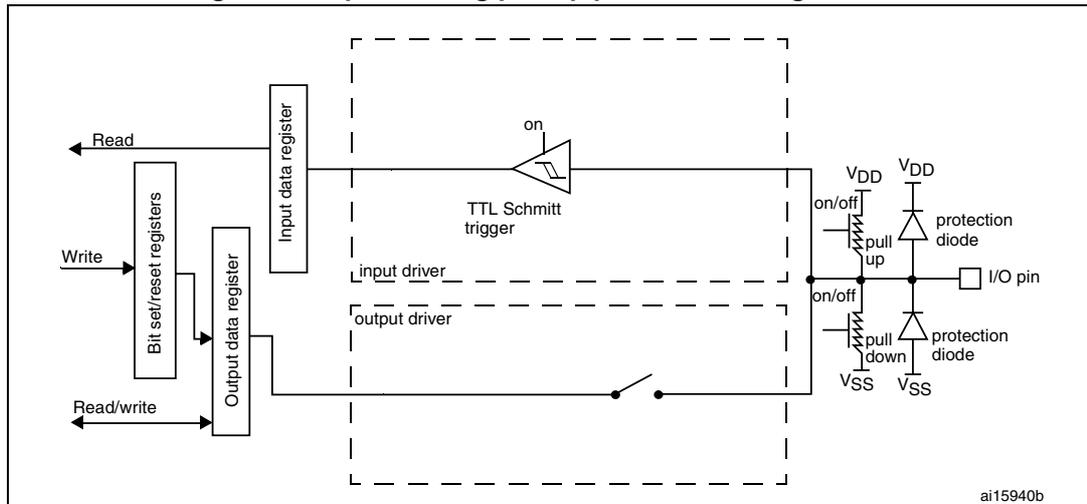
### 8.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx\_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

[Figure 16](#) shows the input configuration of the I/O port bit.

Figure 16. Input floating/pull up/pull down configurations



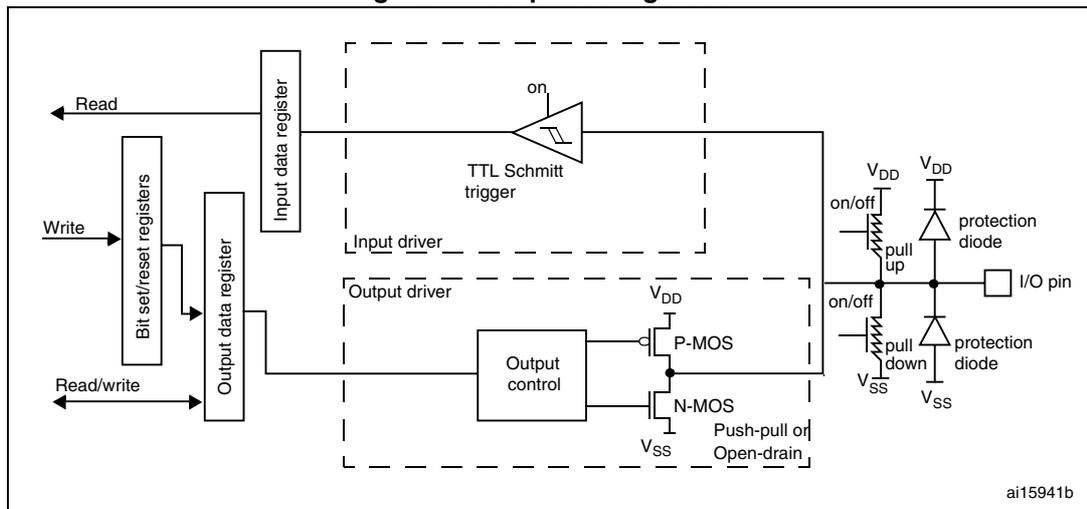
### 8.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx\_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

*Figure 17* shows the output configuration of the I/O port bit.

Figure 17. Output configuration



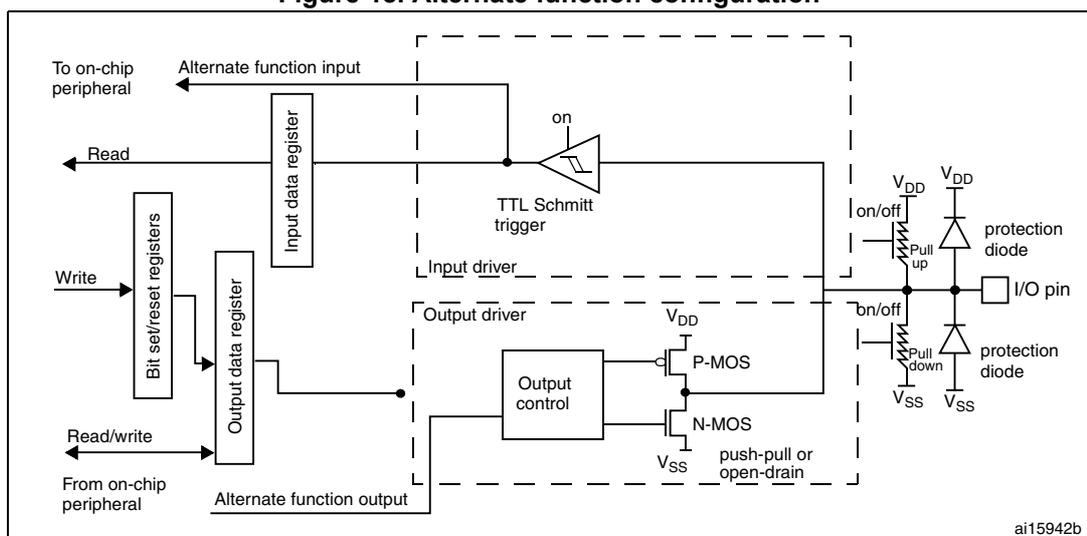
### 8.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx\_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 18 shows the Alternate function configuration of the I/O port bit.

Figure 18. Alternate function configuration



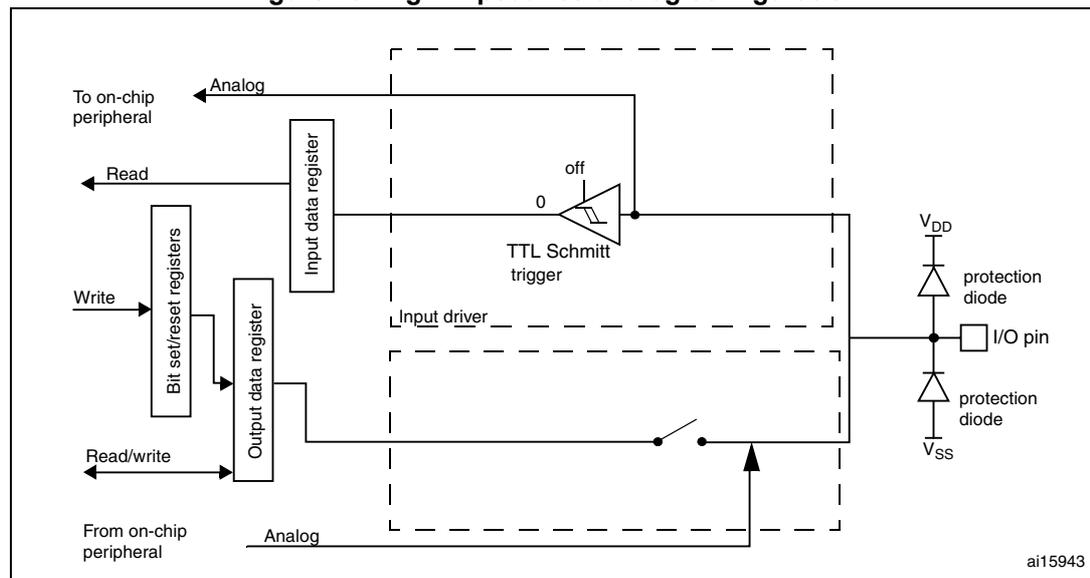
### 8.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware
- Read access to the input data register gets the value “0”

[Figure 19](#) shows the high-impedance, analog-input configuration of the I/O port bit.

**Figure 19. High impedance-analog configuration**



### 8.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC\_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the OSC\_IN or OSC32\_IN pin is reserved for clock input and the OSC\_OUT or OSC32\_OUT pin can still be used as normal GPIO.

### 8.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to [Section 23.3: RTC functional description on page 508](#).

## 8.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to [Table 20](#).

The peripheral registers can be written in word, half word or byte mode.

### 8.4.1 GPIO port mode register (GPIOx\_MODER) (x = A..F)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
r	w	r	w	r	w	r	w	r	w	r	w	r	w	r	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
r	w	r	w	r	w	r	w	r	w	r	w	r	w	r	w

Bits 2y+1:2y **MODERy[1:0]**: Port x configuration bits (y = 0..15)

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

- 00: Input mode (reset state)
- 01: General purpose output mode
- 10: Alternate function mode
- 11: Analog mode

### 8.4.2 GPIO port output type register (GPIOx\_OTYPER) (x = A..F)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
r	w	r	w	r	w	r	w	r	w	r	w	r	w	r	w

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

Bits 15:0 **OTy**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output type.

- 0: Output push-pull (reset state)
- 1: Output open-drain

### 8.4.3 GPIO port output speed register (GPIOx\_OSPEEDR) (x = A..F)

Address offset: 0x08

Reset value:

- 0x0C00 0000 for port A
- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
r/w	r/w	r/w	r/w	r/w	r/w										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3 [1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
r/w	r/w	r/w	r/w	r/w	r/w										

Bits 2y+1:2y OSPEEDR<sub>y</sub>[1:0]: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

x0: Low speed

01: Medium speed

11: High speed

*Note:* Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

### 8.4.4 GPIO port pull-up/pull-down register (GPIOx\_PUPDR) (x = A..F)

Address offset: 0x0C

Reset values:

- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0C00 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rW	rW	rW	rW	rW	rW										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rW	rW	rW	rW	rW	rW										

Bits 2y+1:2y **PUPDRy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

- 00: No pull-up, pull-down
- 01: Pull-up
- 10: Pull-down
- 11: Reserved

### 8.4.5 GPIO port input data register (GPIOx\_IDR) (x = A..F)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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**: Port input data bit (y = 0..15)

These bits are read-only. They contain the input value of the corresponding I/O port.

### 8.4.6 GPIO port output data register (GPIOx\_ODR) (x = A..F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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**: Port output data bit (y = 0..15)

These bits can be read and written by software.

*Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx\_BSRR or GPIOx\_BRR registers (x = A..F).*

### 8.4.7 GPIO port bit set/reset register (GPIOx\_BSRR) (x = A..F)

Address offset: 0x18

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. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets 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. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

### 8.4.8 GPIO port configuration lock register (GPIOx\_LCKR) (x = A, B, and D)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the

LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next reset.

*Note:* A specific write sequence is used to write to the GPIOx\_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															r/w
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
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 16 **LCKK**: Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx\_LCKR register is locked until an MCU reset occurs.

LOCK key write sequence:

WR LCKR[16] = '1' + LCKR[15:0]

WR LCKR[16] = '0' + LCKR[15:0]

WR LCKR[16] = '1' + LCKR[15:0]

RD LCKR

RD LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)

*Note:* During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit will return '1' until the next CPU reset.

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

### 8.4.9 GPIO alternate function low register (GPIOx\_AFRL) (x = A..E)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
r/w	r/w	r/w	r/w												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
r/w	r/w	r/w	r/w												

Bits 31:0 **AFRy[3:0]**: Alternate function selection for port x pin y (y = 0..7)  
 These bits are written by software to configure alternate function I/Os

- AFRy selection:
- |           |                                    |
|-----------|------------------------------------|
| 0000: AF0 | 1000: AF8 (Ports A, B and D only)  |
| 0001: AF1 | 1001: AF9 (Ports A, B and D only)  |
| 0010: AF2 | 1010: AF10 (Ports A, B and D only) |
| 0011: AF3 | 1011: AF11 (Ports A, B and D only) |
| 0100: AF4 | 1100: AF12 (Ports A, B and D only) |
| 0101: AF5 | 1101: AF13 (Ports A, B and D only) |
| 0110: AF6 | 1110: AF14 (Ports A, B and D only) |
| 0111: AF7 | 1111: AF15 (Ports A, B and D only) |

### 8.4.10 GPIO alternate function high register (GPIOx\_AFRH) (x = A..F)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
r/w	r/w	r/w	r/w												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
r/w	r/w	r/w	r/w												

Bits 31:0 **AFRy[3:0]**: Alternate function selection for port x pin y (y = 8..15)  
 These bits are written by software to configure alternate function I/Os

- AFRy selection:
- |           |                                    |
|-----------|------------------------------------|
| 0000: AF0 | 1000: AF8 (Ports A, B and D only)  |
| 0001: AF1 | 1001: AF9 (Ports A, B and D only)  |
| 0010: AF2 | 1010: AF10 (Ports A, B and D only) |
| 0011: AF3 | 1011: AF11 (Ports A, B and D only) |
| 0100: AF4 | 1100: AF12 (Ports A, B and D only) |
| 0101: AF5 | 1101: AF13 (Ports A, B and D only) |
| 0110: AF6 | 1110: AF14 (Ports A, B and D only) |
| 0111: AF7 | 1111: AF15 (Ports A, B and D only) |



**8.4.11 GPIO port bit reset register (GPIOx\_BRR) (x = A..F)**

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
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. A read to these bits returns the value 0x0000

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

### 8.4.12 GPIO register map

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

**Table 20. 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	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	GPIOB_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[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	1	0	1	0	0	0	0	0	0	0
0x00	GPIOx_MODER (where x = C..F)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPIOx_OTYPER (where x = A..F)	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0										
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOA_OSPEEDR	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOB_OSPEEDR	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[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	1	1	0	0	0	0	0	0
0x08	GPIOx_OSPEEDR (where x = C..F)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOA_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 20. GPIO 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
0x0C	GPIOB_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[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	1	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (where x = C..F)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	GPIOx_IDR (where x = A..F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	GPIOx_ODR (where x = A..F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (where x = A..F)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (where x = A, B, D)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFR1 (where x = A..F)	AFR7[3:0]			AFR6[3:0]			AFR5[3:0]			AFR4[3:0]			AFR3[3:0]			AFR2[3:0]			AFR1[3:0]			AFR0[3:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (where x = A..F)	AFR15[3:0]			AFR14[3:0]			AFR13[3:0]			AFR12[3:0]			AFR11[3:0]			AFR10[3:0]			AFR9[3:0]			AFR8[3:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (where x = A..F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2](#) for the register boundary addresses.



## 9 System configuration controller (SYSCFG)

The devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Enabling/disabling I<sup>2</sup>C Fast Mode Plus on some I/O ports
- Remapping some DMA trigger sources from TIM16 and TIM17, USART1, and ADC to different DMA channels
- Remapping the memory located at the beginning of the code area
- Managing the external interrupt line connection to the GPIOs
- Managing robustness feature

### 9.1 SYSCFG registers

#### 9.1.1 SYSCFG configuration register 1 (SYSCFG\_CFGR1)

This register is used for specific configurations on memory remap.

Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.

After reset these bits take the value selected by the BOOT pin (BOOT0) and by the option bite (nBOOT1).

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the BOOT0 pin and nBOOT1 option bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FPU_IE[5:0]						Res.	VBAT_MON	Res.	Res.	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP
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
TIM18_DAC2_OUT1_DMA_RMP	TIM7_DAC1_OUT2_DMA_RMP	TIM6_DAC1_OUT1_DMA_RMP	TIM17_DMA_RMP	TIM16_DMA_RMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM_MODE	
rw	rw		rw	rw										rw	rw

- Bits 31:26 **FPU\_IE[5:0]**: Floating point unit interrupts enable bits.
- FPU\_IE[5]: Inexact interrupt enable
  - FPU\_IE[4]: Input denormal interrupt enable
  - FPU\_IE[3]: Overflow interrupt enable
  - FPU\_IE[2]: underflow interrupt enable
  - FPU\_IE[1]: Divide-by-zero interrupt enable
  - FPU\_IE[0]: Invalid operation interrupt enable
- Bit 25 Reserved, must be kept at reset value.
- Bit 24 **VBAT\_MON**: V<sub>BAT</sub> monitoring enable
- This bit is set and cleared by software. When it is set, it enables the power switch to deliver V<sub>BAT</sub> voltage on ADC channel 18 input.
- Bits 23:22 Reserved, must be kept at reset value.
- Bit 21 **I2C2\_FMP**: I2C2 Fast Mode Plus (Fm+) driving capability activation bit, whatever the AFI/AFO mapping.
- This bit is set and cleared by software. When it is set, the Fm+ mode is enabled on I2C2 pins selected through IOPORT control registers AF selection bits. This bit is OR-ed with I2C\_PBx\_FMP bits.
- Bit 20 **I2C1\_FMP**: I2C1 Fast Mode Plus (Fm+) driving capability activation bit, whatever the AFI/AFO mapping.
- This bit is set and cleared by software. When it is set, the Fm+ mode is enabled on I2C1 pins selected through IOPORT control registers AF selection bits. This bit is OR-ed with I2C\_PBx\_FMP bits.
- Bits 19:16 **I2C\_PBx\_FMP**: Fast Mode Plus (Fm+) driving capability activation bits.
- These bits are set and cleared by software. Each bit enables I<sup>2</sup>C Fm+ mode for PB6, PB7, PB8, and PB9 I/Os.
  - 0: PBx pin operates in standard mode.
  - 1: I<sup>2</sup>C Fm+ mode enabled on PBx pin, and the Speed control is bypassed.
- Bit 15 **TIM18\_DAC2\_OUT1\_DMA\_RMP**: TIM18 and DAC2\_OUT1 DMA request remapping bit
- This bit is set and cleared by software. It controls the remapping of TIM18 and DAC2\_OUT1 DMA request.
  - 0: No remap (TIM18 and DAC2\_OUT1 DMA requests mapped on DMA2 channel 5)
  - 1: Remap (TIM18 and DAC2\_OUT1 DMA requests mapped on DMA1 channel 5)
- Bit 14 **TIM7\_DAC1\_OUT2\_DMA\_RMP**: TIM7 and DAC1\_OUT2 DMA request remapping bit
- This bit is set and cleared by software. It controls the remapping of TIM7 and DAC1\_OUT2 DMA request.
  - 0: No remap (TIM7 and DAC1\_OUT2 DMA requests mapped on DMA2 channel 4)
  - 1: Remap (TIM7 and DAC1\_OUT2 DMA requests mapped on DMA1 channel 4)
- Bit 13 **TIM6\_DAC1\_OUT1\_DMA\_RMP**: TIM6 and DAC1\_OUT1 DMA request remapping bit
- This bit is set and cleared by software. It controls the remapping of TIM6 and DAC1\_OUT1 DMA request.
  - 0: No remap (TIM7 and DAC1\_OUT1 DMA requests mapped on DMA2 channel 3)
  - 1: Remap (TIM7 and DAC1\_OUT1 DMA requests mapped on DMA1 channel 3)
- Bit 12 **TIM17\_DMA\_RMP**: TIM17 DMA request remapping bit
- This bit is set and cleared by software. It controls the remapping of TIM17 DMA request.
  - 0: No remap (TIM17\_CH1 and TIM17\_UP DMA requests mapped on DMA channel 1)
  - 1: Remap (TIM17\_CH1 and TIM17\_UP DMA requests mapped on DMA channel 2)

Bit 11 TIM16\_DMA\_RMP: TIM16 DMA request remapping bit

This bit is set and cleared by software. It controls the remapping of TIM16 DMA request.

0: No remap (TIM16\_CH1 and TIM16\_UP DMA requests mapped on DMA channel 3)

1: Remap (TIM16\_CH1 and TIM16\_UP DMA requests mapped on DMA channel 4)

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **MEM\_MODE**: Memory mapping selection bits

This bit is set and cleared by software. It controls the memory internal mapping at address 0x0000 0000. After reset these bits take on the memory mapping selected by BOOT0 pin and nBOOT1 option bit.

x0: Main Flash memory mapped at 0x0000 0000

01: System Flash memory mapped at 0x0000 0000

11: Embedded SRAM mapped at 0x0000 0000

### 9.1.2 SYSCFG external interrupt configuration register 1 (SYSCFG\_EXTICR1)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
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												

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

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

These bits are written by software to select the source input for the EXTIx external interrupt.

x000: PA[x] pin

x001: PB[x] pin

x010: PC[x] pin

x011: PD[x] pin

x100: PE[x] pin

x101: PF[x] pin (x = 0 to 2)

other configurations: reserved

*Note:* Some of the I/O pins mentioned in the above register may not be available on small packages.

### 9.1.3 SYSCFG external interrupt configuration register 2 (SYSCFG\_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
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]			
r/w	r/w	r/w	r/w												

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

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

These bits are written by software to select the source input for the EXTIx external interrupt.

- x000: PA[x] pin
- x001: PB[x] pin
- x010: PC[x] pin
- x011: PD[x] pin
- x100: PE[x] pin
- x101: PF[x] pin (x = 4, 6, 7)
- other configurations: reserved

*Note:* Some of the I/O pins mentioned in the above register may not be available on small packages.

### 9.1.4 SYSCFG external interrupt configuration register 3 (SYSCFG\_EXTICR3)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w



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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration bits (x = 8 to 11)

These bits are written by software to select the source input for the EXTIx external interrupt.

- x000: PA[x] pin
- x001: PB[x] pin (x= 8 to 10)
- x010: PC[x] pin
- x011: PD[x] pin
- x100: PE[x] pin
- x101: PF[x] pin (x = 9, 10)
- other configurations: reserved

*Note:* Some of the I/O pins mentioned in the above register may not be available on small packages.

### 9.1.5 SYSCFG external interrupt configuration register 4 (SYSCFG\_EXTICR4)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.												
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												

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

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration bits (x = 12 to 15)

These bits are written by software to select the source input for the EXTIx external interrupt.

- x000: PA[x] pin
- x001: PB[x] pin (x= 14, 15)
- x010: PC[x] pin
- x011: PD[x] pin
- x100: PE[x] pin
- other configurations: reserved

*Note:* Some of the I/O pins mentioned in the above register may not be available on small packages.

### 9.1.6 SYSCFG configuration register 2 (SYSCFG\_CFGR2)

Address offset: 0x18

System reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SRAM_PEF	Res.	Res.	Res.	Res.	Res.	PVD_LOCK	SRAM_PARITY_LOCK	LOCKUP_LOCK						
							rc_w1						rW	rW	rW

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

Bit 8 **SRAM\_PEF**: SRAM parity flag

This bit is set by hardware when an SRAM parity error is detected. It is cleared by software by writing '1'.

- 0: No SRAM parity error detected
- 1: SRAM parity error detected

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

Bit 2 **PVD\_LOCK**: PVD lock enable bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the PVD connection to TIM15/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR\_CR register.

- 0: PVD interrupt disconnected from TIM15/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application.
- 1: PVD interrupt connected to TIM15/16/17 Break input, PVDE and PLS[2:0] bits are read only.

Bit 1 **SRAM\_PARITY\_LOCK**: SRAM parity lock bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the SRAM parity error signal connection to TIM15/16/17 Break input.

- 0: SRAM parity error disconnected from TIM15/16/17 Break input
- 1: SRAM parity error connected to TIM15/16/17 Break input

Bit 0 **LOCKUP\_LOCK**: Cortex<sup>®</sup>-M4F LOCKUP enable bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the connection of Cortex<sup>®</sup>-M4F LOCKUP (Hardfault) output to TIM15/16/17 Break input.

- 0: Cortex<sup>®</sup>-M4F LOCKUP output disconnected from TIM15/16/17 Break input
- 1: Cortex<sup>®</sup>-M4F LOCKUP output connected to TIM15/16/17 Break input



## 10 Direct memory access controller (DMA)

### 10.1 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.

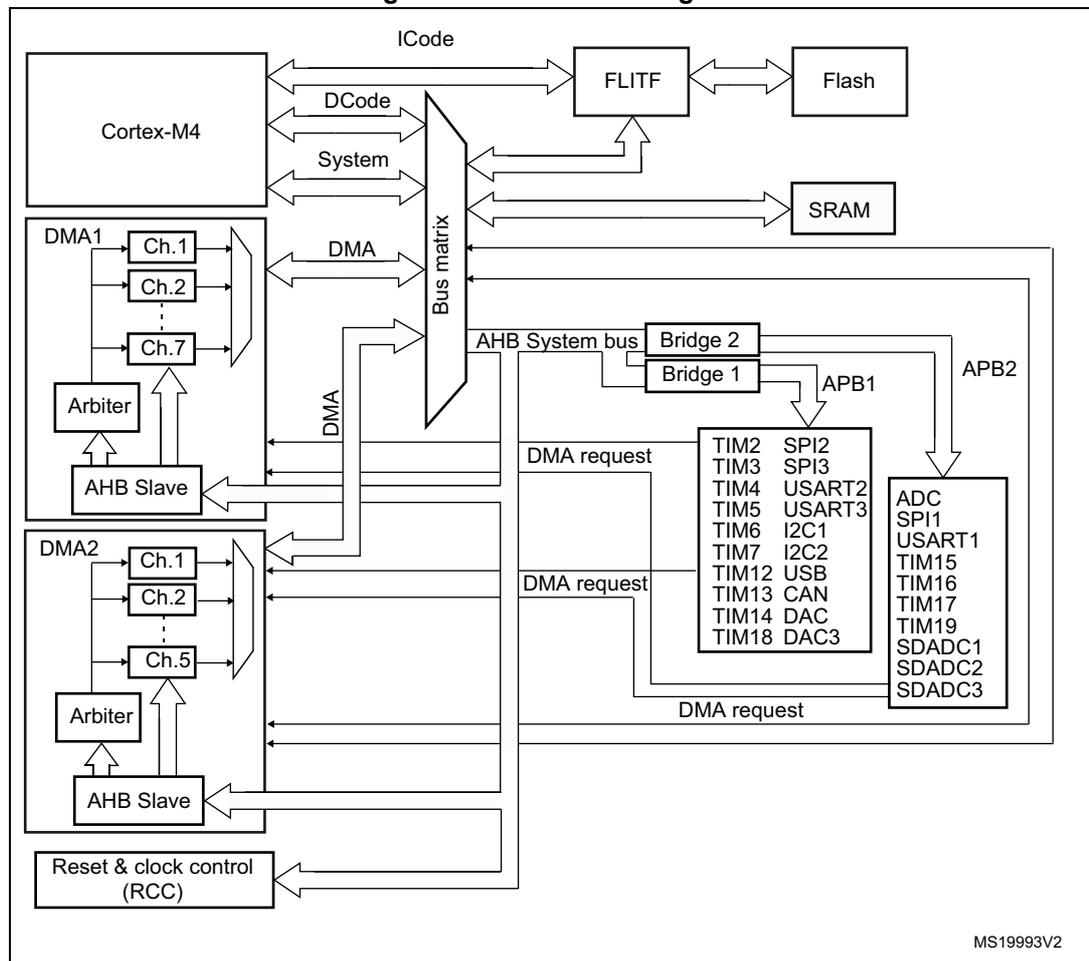
The two DMA controllers have 12 channels in total, each dedicated to managing memory access requests from one or more peripherals. Each has an arbiter for handling the priority between DMA requests.

### 10.2 DMA main features

- 12 independently configurable channels (requests)
- Each channel 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, APB and AHB peripherals as source and destination
- Programmable number of data to be transferred: up to 65535

The block diagram is shown in the following figure.

Figure 20. DMA block diagram



### 10.3 DMA implementation

This manual describes the full set of features implemented in DMA1. DMA2 supports a smaller number of channels, but is otherwise identical to DMA1.

Table 22. DMA implementation

Feature	DMA1	DMA2
Number of DMA channels	7	5

## 10.4 DMA functional description

The DMA controller performs direct memory transfer by sharing the system bus with the Cortex<sup>®</sup>-M4 with FPU 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.

### 10.4.1 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 de-asserted 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.

### 10.4.2 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.

### 10.4.3 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 non-circular 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 channel x (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.

**10.4.4 Programmable data width, data alignment and endians**

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in [Table 23: Programmable data width & endian behavior \(when bits PINC = MINC = 1\)](#).

**Table 23. 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



**Table 23. Programmable data width & endian behavior (when bits PINC = MINC = 1) (continued)**

Source port width	Destination port width	Number of data items to transfer (NDT)	Source content: address / data	Transfer operations	Destination content: address / data
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[15:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B5B4[15:0] @0x2 3: READ BBBAB9B8[31:0] @0x8 then WRITE B9B8[15:0] @0x4 4: READ BFBEBDBC[31:0] @0xC then WRITE BDBC[15:0] @0x6	@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”.

**10.4.5 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.



### 10.4.6 DMA 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 24. DMA interrupt requests**

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

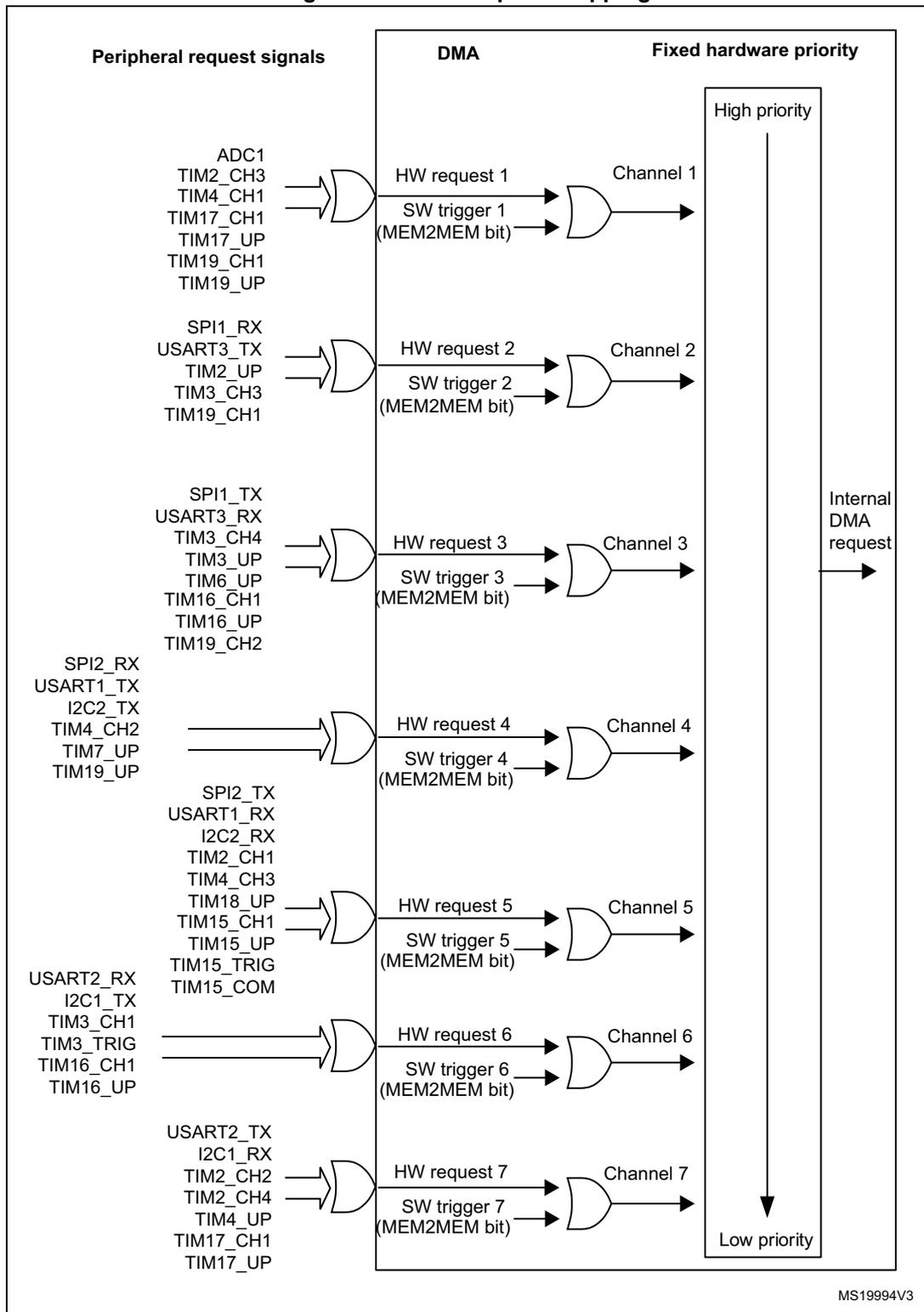
## 10.4.7 DMA request mapping

### DMA1 controller

The hardware requests from the peripherals (TIMx, ADC1, DACx, SPIx, I2Cx, and USARTx) are simply logically ORed before entering the DMA1. This means that on one channel, only one request must be enabled at a time. Refer to [Figure 21: DMA1 request mapping](#).

The peripheral DMA1 requests can be independently activated/de-activated by programming the DMA1 control bit in the registers of the corresponding peripheral.

Figure 21. DMA1 request mapping



1. DMA request mapped on this DMA channel only if the corresponding remapping bit is cleared in the SYSCFG\_CFGR1 register. For more details, please refer to [Section 9.1.1: SYSCFG configuration register 1 \(SYSCFG\\_CFGR1\) on page 155](#).

Table 25 lists the DMA1 requests for each channel.

**Table 25. Summary of DMA1 requests for each channel**

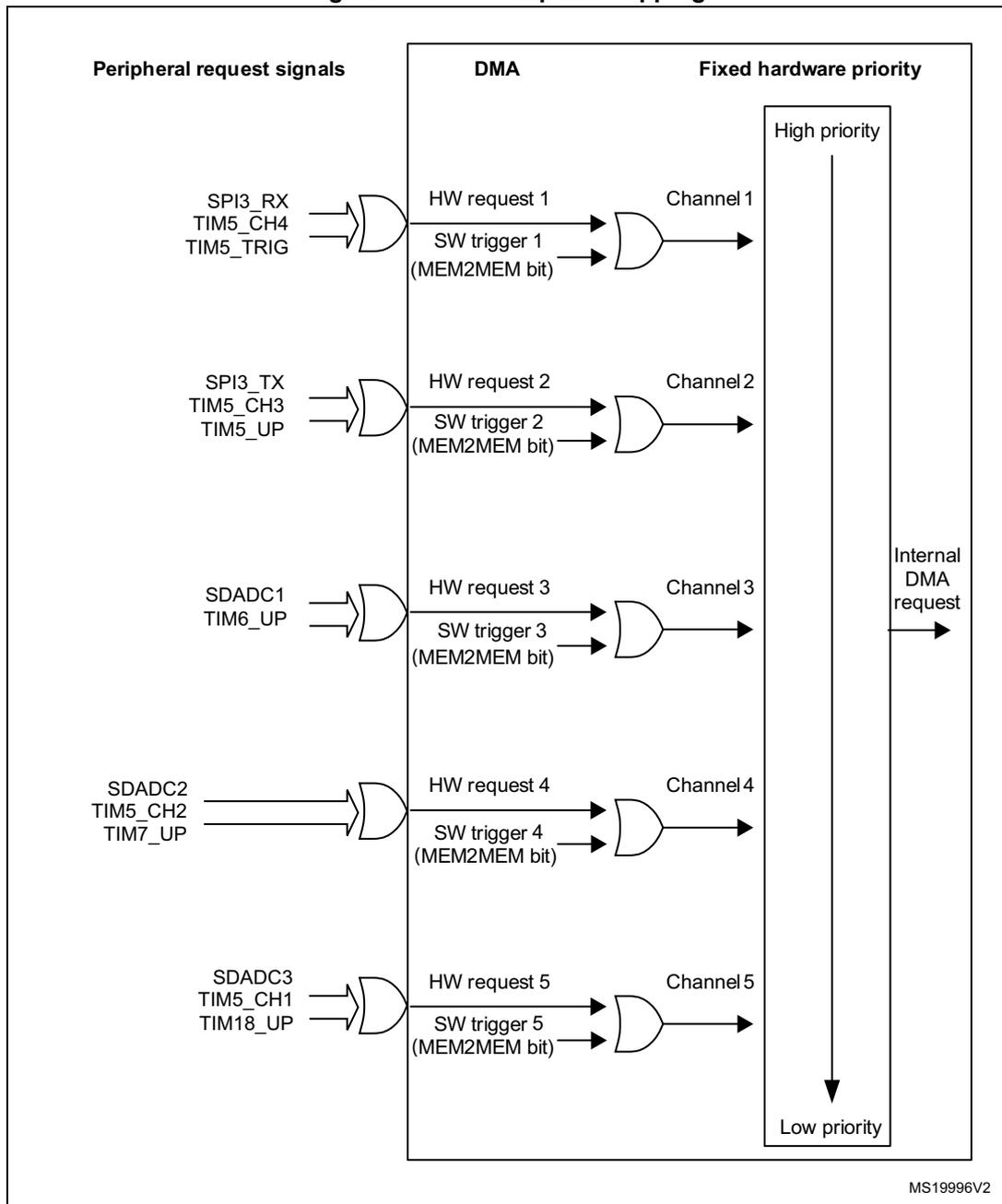
Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
<b>ADC1</b>	ADC1						
<b>SPI</b>		SPI1_RX	SP1_TX	SPI2_RX	SPI2_TX		
<b>USART</b>		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
<b>I2C</b>				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
<b>TIM2</b>	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
<b>TIM3</b>		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
<b>TIM4</b>	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP
<b>TIM6 / DAC channel 1</b>			TIM6_UP DAC1_CH1				
<b>TIM7 / DAC channel 2</b>				TIM7_UP DAC1_CH2			
<b>TIM18 / DAC channel 3</b>					TIM18_UP DAC2_CH1		
<b>TIM15</b>					TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM		
<b>TIM16</b>			TIM16_CH1 TIM16_UP			TIM16_CH1 TIM16_UP	
<b>TIM17</b>	TIM17_CH1 TIM17_UP						TIM17_CH1 TIM17_UP
<b>TIM19</b>	TIM19_CH3 TIM19_CH4	TIM19_CH1	TIM19_CH2	TIM19_UP			

**DMA2 controller**

The hardware requests from the peripherals (TIMx, SDADCx, DAC and SPI) are simply logically ORed before entering the DMA2. This means that on one channel, only one request must be enabled at a time. Refer to [Figure 22: DMA2 request mapping](#).

The peripheral DMA2 requests can be independently activated/de-activated by programming the DMA2 control bit in the registers of the corresponding peripheral.

Figure 22. DMA2 request mapping



1. DMA request mapped on this DMA channel only if the corresponding remapping bit is cleared in the SYSCFG\_CFGR1 register. For more details, please refer to [Section 9.1.1: SYSCFG configuration register 1 \(SYSCFG\\_CFGR1\) on page 155](#).

Table 26 lists the DMA2 requests for each channel.

**Table 26. Summary of DMA2 requests for each channel**

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
<b>SDADC</b>			SDADC1	SDADC2	SDADC3
<b>SPI</b>	SPI3_RX	SPI3_TX			
<b>TIM5</b>	TIM5_CH4 TIM5_TRIG	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
<b>TIM6 / DAC channel 1</b>			TIM6_UP DAC1_CH1		
<b>TIM7 / DAC channel 2</b>				TIM7_UP DAC1_CH2	
<b>TIM18 / DAC channel 3</b>					TIM18_UP DAC2_CH1

## 10.5 DMA registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by bytes (8-bit), half-words (16-bit) or words (32-bit).

### 10.5.1 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
Res.	Res.	Res.	Res.	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

### 10.5.2 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
Res.	Res.	Res.	Res.	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, **CTEIFx**: Channel x transfer error clear (x = 1 ..7)

11, 7, 3 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, **CHTIFx**: Channel x half transfer clear (x = 1 ..7)

10, 6, 2 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, **CTCIFx**: Channel x transfer complete clear (x = 1 ..7)

9, 5, 1 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, **CGIFx**: Channel x global interrupt clear (x = 1 ..7)

8, 4, 0 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

**10.5.3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

### 10.5.4 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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.

### 10.5.5 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
PA [31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA [15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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.

**10.5.6 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
MA [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
MA [15:0]															
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.

### 10.5.7 DMA register map

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

**Table 27. 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
0x00	DMA_ISR	Res.	Res.	Res.	Res.	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
0x04	DMA_IFCR	Res.	Res.	Res.	Res.	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
0x08	DMA_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]	MSIZE [1:0]												
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DMA_CNDTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	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
0x14	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
0x1C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]	MSIZE [1:0]												
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	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
0x28	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
0x30	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]	MSIZE [1:0]												
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	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
0x3C	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
0x44	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]	MSIZE [1:0]												
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 27. 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	
0x4C	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	0
0x50	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	0
0x54	Reserved																																	
0x58	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]	MSIZE [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	
0x5C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	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	0
0x64	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	0
0x68	Reserved																																	
0x6C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]	MSIZE [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	0
0x70	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	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	0
0x78	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	0
0x7C	Reserved																																	
0x80	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL [1:0]	MSIZE [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	0
0x84	DMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	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	0
0x8C	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	0
0x90	Reserved																																	

Refer to [Section 2.2.2](#) for the register boundary addresses.



# 11 Interrupts and events

## 11.1 Nested vectored interrupt controller (NVIC)

### 11.1.1 NVIC main features

- 1 non-maskable interrupt line (NMI)
- 64 maskable interrupt channels
- 16 programmable priority levels (2 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of System Control Registers

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to *PM0214 programming manual*.

### 11.1.2 SysTick calibration value register

The SysTick calibration value is set to 9000, which gives a reference time base of 1 ms with the SysTick clock set to 9 MHz (max  $f_{HCLK}/8$ ).

### 11.1.3 Interrupt and exception vectors

[Table 28](#) is the vector table for STM32F37xx devices.

**Table 28. List of vectors**

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
	-1	fixed	HardFault	Hardware fault	0x0000 000C
	0	fixed	MemManage	MPU fault	0x0000 0010
	1	settable	BusFault	Prefetch fault or memory access error	0x0000 0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018

Table 28. List of vectors (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
	4	settable	DebugMonitor	Debug monitor	0x0000 0030
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	Power voltage detector through EXTI line detection interrupt	0x0000 0044
2	9	settable	TAMP	Tamper and timestamp through EXTI19 line	0x0000 0048
3	10	settable	RTC_WKUP	RTC	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 Line 0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line 1 interrupt	0x0000 005C
8	15	settable	EXTI2_TS	EXTI Line 2 and routing interface interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line 3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line 4 interrupt	0x0000 0068
11	18	settable	DMA1_CH1	DMA1 channel 1 interrupt	0x0000 006C
12	19	settable	DMA1_CH2	DMA1 channel 2 interrupt	0x0000 0070
13	20	settable	DMA1_CH3	DMA1 channel 3 interrupt	0x0000 0074
14	21	settable	DMA1_CH4	DMA1 channel 4 interrupt	0x0000 0078
15	22	settable	DMA1_CH5	DMA1 channel 5 interrupt	0x0000 007C
16	23	settable	DMA1_CH6	DMA1 channel 6 interrupt	0x0000 0080
17	24	settable	DMA1_CH7	DMA1 channel 7 interrupt	0x0000 0084
18	25	settable	ADC1	ADC1 interrupt	0x0000 0088
19	26	settable	CAN TX	CAN_TX interrupt	0x0000 008C
20	27	settable	CAN RXD	CAN_RXD interrupt	0x0000 0090
21	28	settable	CAN RXI	CAN_RXI interrupt	0x0000 0094
22	29	settable	CAN SCE	CAN_SCE interrupt	0x0000 0098

Table 28. List of vectors (continued)

Position	Priority	Type of priority	Acronym	Description	Address
23	30	settable	EXTI5_9	EXTI Line[9:5] interrupts	0x0000 009C
24	31	settable	TIM15	Timer 15 global interrupt	0x0000 00A0
25	32	settable	TIM16	Timer 16 global interrupt	0x0000 00A4
26	33	settable	TIM17	Timer 17 global interrupt	0x0000 00A8
27	34	settable	TIM18_DAC2	Timer 18 global interrupt/DAC2 underrun interrupt	0x0000 00AC
28	35	settable	TIM2	Timer 2 global interrupt	0x0000 00B0
29	36	settable	TIM3	Timer 3 global interrupt	0x0000 00B4
30	37	settable	TIM4	Timer 4 global interrupt	0x0000 00B8
31	38	settable	I2C1_EV	I2C1_EV global interrupt/EXTI Line[3:2] interrupts	0x0000 00BC
32	39	settable	I2C1_ER	I2C1_ER	0x0000 00C0
33	40	settable	I2C2_EV	I2C2_EV global interrupt/EXTI Line[4:2] interrupts	0x0000 00C4
34	41	settable	I2C2_ER	I2C2_ER	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/EXTI25 (USART1 wakeup event)	0x0000 00D4
38	45	settable	USART2	USART2 global interrupt/EXTI26 (USART2 wakeup event)	0x0000 00D8
39	46	settable	USART3	USART3 global interrupt/EXTI28 (USART3 wakeup event)	0x0000 00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000 00E0
41	48	settable	RTC_ALARM_IT	RTC alarm interrupt	0x0000 00E4
42	49	settable	CEC	CEC interrupt	0x0000 00E8
43	50	settable	TIM12	Timer 12 global interrupt	0x0000 00EC
44	51	settable	TIM13	Timer 13 global interrupt	0x0000 00F0
45	52	settable	TIM14	Timer 14 global interrupt	0x0000 00F4
46-49	53-56	Reserved			0x0000 00F8-0x0000 0104
50	57	settable	TIM5	Timer 5 global interrupt	0x0000 0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000 010C

Table 28. List of vectors (continued)

Position	Priority	Type of priority	Acronym	Description	Address
52-53	59-60	Reserved			0x0000 0110-0x0000 0114
54	61	settable	TIM6_DAC1	Timer 6 global interrupt/DAC1 underrun interrupt	0x0000 0118
55	62	settable	TIM7	Timer 7 global interrupt	0x0000 011C
56	63	settable	DMA2_CH1	DMA2 channel 1 interrupt	0x0000 0120
57	64	settable	DMA2_CH2	DMA2 channel 2 interrupt	0x0000 0124
58	65	settable	DMA2_CH3	DMA2 channel 3 interrupt	0x0000 0128
59	66	settable	DMA2_CH4	DMA2 channel 4 interrupt	0x0000 012C
60	67	settable	DMA2_CH5	DMA2 channel 5 interrupt	0x0000 0130
61	68	settable	SDADC1	ADC sigma delta 1 (SDADC1) global interrupt	0x0000 0134
62	69	settable	SDADC2	ADC sigma delta 2 (SDADC2) global interrupt	0x0000 0138
63	70	settable	SDADC3	ADC sigma delta 1 (SDADC3) global interrupt	0x0000 013C
64	71	settable	COMP1_2	Comparator 1/comparator 2 global interrupts (EXTI21/EXTI22)	0x0000 0140
65-73	72-80	Reserved			0x0000 0144-0x0000 0164
74	81	settable	USB_HP	USB high priority interrupt	0x0000 0168
75	82	settable	USB_LP	USB low priority interrupt	0x0000 016C
76	83	settable	USB_WAKEUP	USB wakeup interrupt	0x0000 0170
77	84	Reserved			0x0000 0174
78	85	settable	TIM19	Timer 19 global interrupt	0x0000 0178
79-80	86-87	Reserved			0x0000 017C-0x0000 0180
81	88	settable	FPU	Floating point unit interrupt	0x0000 0184

## 11.2 Extended interrupts and events controller (EXTI)

The extended interrupts and events controller (EXTI) manages the external and internal asynchronous events/interrupts and generates the event request to the CPU/Interrupt Controller and a wake-up request to the Power Manager.

The EXTI allows the management of up to 29 external/internal event line (21 external event lines and 8 internal event lines).

The active edge of each external interrupt line can be chosen independently, whilst for internal interrupt the active edge is always the rising one. An interrupt could be left pending: in case of an external one, a status register is instantiated and indicates the source of the interrupt; an event is always a simple pulse and it's used for triggering the core Wake-up. For internal interrupts, the pending status is assured by the generating IP, so no need for a specific flag. Each input line can be masked independently for interrupt or event generation, in addition the internal lines are sampled only in STOP mode. This controller allows also to emulate the (only) external events by software, multiplexed with the corresponding hardware event line, by writing to a dedicated register.

### 11.2.1 Main features

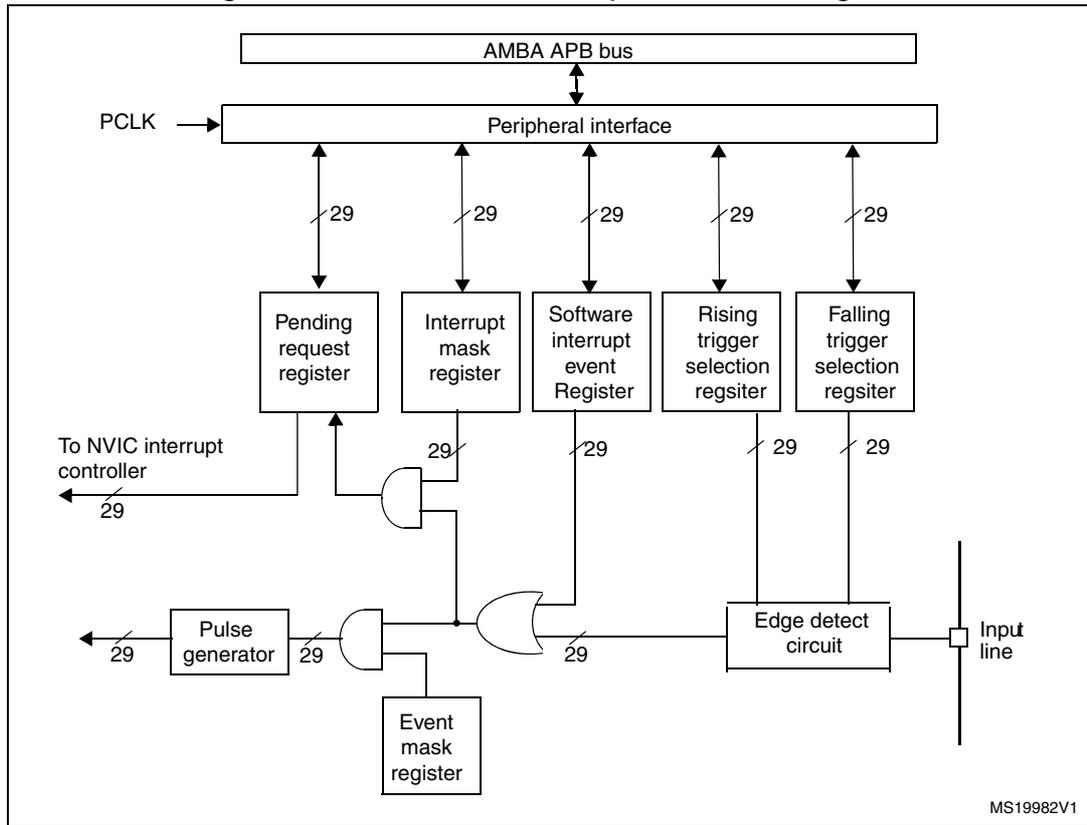
The EXTI main features are the following:

- support generation of up to 29 event/interrupt requests;
- Independent configuration of each line as an external or an internal event requests;
- Independent mask on each event/interrupt line
- Automatic disable of internal lines when system is not in STOP mode
- Independent trigger for external event/interrupt line
- Dedicated status bit for external interrupt line;
- Emulation for all the external event requests.

### 11.2.2 Block diagram

The extended interrupt/event block diagram is shown in [Figure 23](#).

Figure 23. EXTI extended interrupt/event block diagram



### 11.2.3 Wakeup event management

The STM32F37xx 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<sup>®</sup>-M4 with FPU System Control register. When the MCU resumes from WFE, the EXTI peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or by 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.

### 11.2.4 Asynchronous Internal Interrupts

Some communication peripherals (UART, I2C, CEC) are able to generate events when the system is in run mode and also when the system is in stop mode allowing to wake up the system from stop mode.

To accomplish this, the peripheral is asked to generate both a synchronized (to the system clock, e.g. APB clock) and an asynchronous version of the event.

## 11.2.5 Functional description

For the external interrupt lines, 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.

For the internal interrupt lines, the active edge is always the rising edge, the interrupt is enabled by default in the interrupt mask register and there is no corresponding pending bit 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.

For the external lines, an interrupt/event request can also be generated by software by writing a '1' in the software interrupt/event register.

*Note: The interrupts or events associated to the internal lines can be triggered only when the system is in STOP mode. If the system is still running, no interrupt/event is generated.*

### Hardware interrupt selection

To configure a line as interrupt source, use the following procedure:

- Configure the corresponding mask bit in the EXTI\_IMR register.
- Configure the Trigger Selection bits of the Interrupt line (EXTI\_RTISR and EXTI\_FTISR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the EXTI so that an interrupt coming from one of the EXTI line can be correctly acknowledged.

### Hardware event selection

To configure a line as event source, use the following procedure:

- Configure the corresponding mask bit in the EXTI\_EMR register.
- Configure the Trigger Selection bits of the Event line (EXTI\_RTISR and EXTI\_FTISR)

### Software interrupt/event selection

Any of the external lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

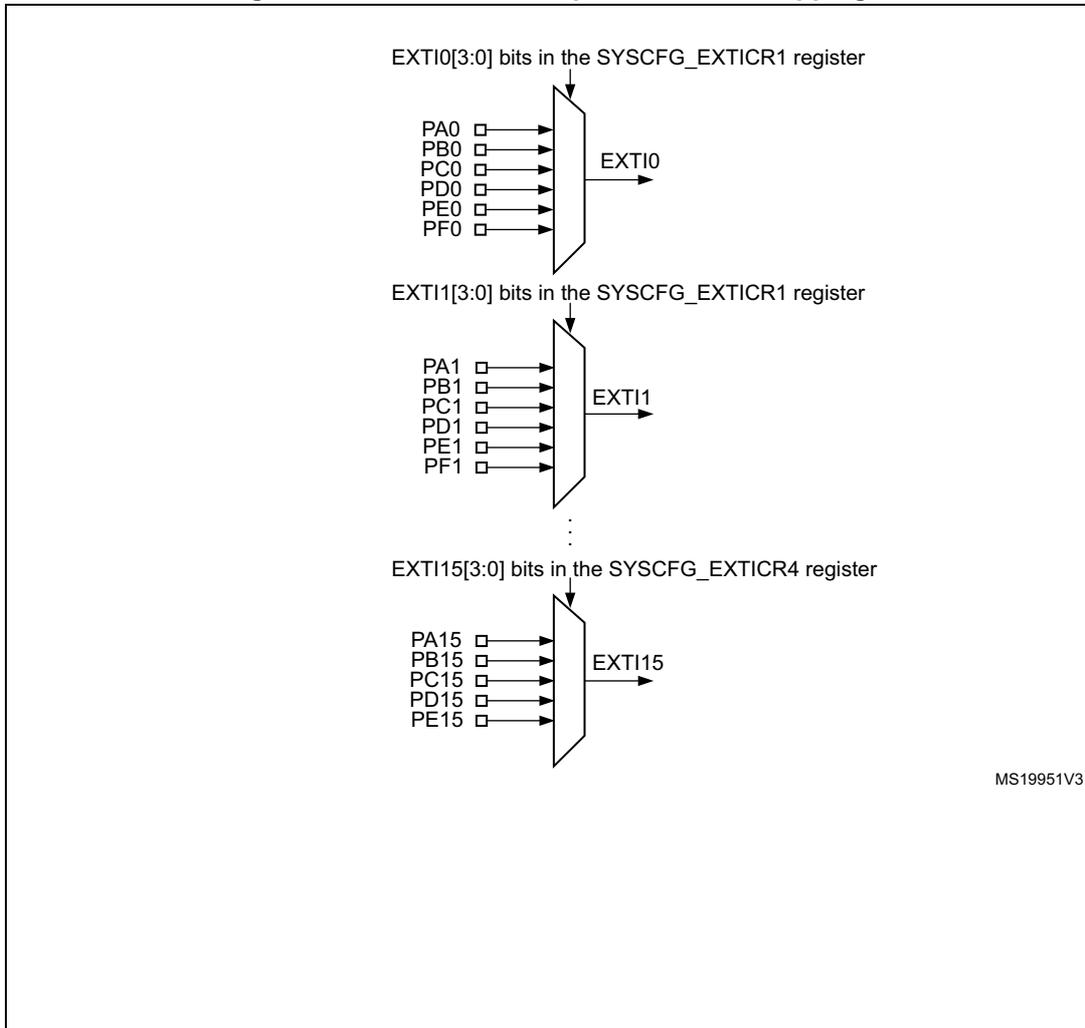
- Configure the corresponding mask bit (EXTI\_IMR, EXTI\_EMR)
- Set the required bit of the software interrupt register (EXTI\_SWIER)

### 11.2.6 External and internal interrupt/event line mapping

In the STM32F37xx, 29 interrupt/event lines are available: 8 lines are internal (including the reserved ones) and the remaining 21 lines are external.

The GPIOs are connected to the 16 external interrupt/event lines in the following manner:

**Figure 24. Extended interrupt/event GPIO mapping**



The remaining lines are connected as follow:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to USB FS wakeup event
- EXTI line 19 is connected to RTC tamper and Timestamps
- EXTI line 20 is connected to RTC wakeup
- EXTI line 21 is connected to Comparator 1 output
- EXTI line 22 is connected to Comparator 2 output
- EXTI line 23 is connected to I2C1 wakeup
- EXTI line 24 is connected to I2C2 wakeup
- EXTI line 25 is connected to USART1 wakeup
- EXTI line 26 is connected to USART2 wakeup
- EXTI line 27 is connected to CEC wakeup
- EXTI line 28 is connected to USART3 wakeup.

*Note: EXTI lines 18, 20, 23, 24, 25, 26, 27, and 28 are internal.*

### 11.3 EXTI registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

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

#### 11.3.1 Interrupt mask register (EXTI\_IMR)

Address offset: 0x00

Reset value: 0x1F80 0000 (See note below)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	MR28	MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20	MR19	MR18	MR17	MR16
			r/w												
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
r/w															

Bits 31:29 Reserved, must be kept at reset value (0).

Bits 28:0 **MRx**: Interrupt Mask on external/internal line x

0: Interrupt request from Line x is masked

1: Interrupt request from Line x is not masked

*Note:* The reset value for the internal lines (23, 24, 25, 26, 27 and 28) is set to '1' in order to enable the interrupt by default.

#### 11.3.2 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
Res.	Res.	Res.	MR28	MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20	MR19	MR18	MR17	MR16
			r/w												
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
r/w															

Bits 31:29 Reserved, must be kept at reset value (0).

Bits 28:0 **MRx**: Event mask on external/internal line x

0: Event request from Line x is masked

1: Event request from Line x is not masked

### 11.3.3 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
Res.	TR19	Res.	TR17	TR16											
												r/w		r/w	r/w
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
r/w															

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TR19**: Rising trigger event configuration bit of line 19  
 0: Rising trigger disabled (for Event and Interrupt) for input line  
 1: Rising trigger enabled (for Event and Interrupt) for input line.

Bits 18 Reserved, must be kept at reset value.

Bits 17:0 **TRx**: Rising trigger event configuration bit of line x (x = 17 to 0)  
 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 an external interrupt line occurs during a write operation to the EXTI\_RTISR register, the pending bit is not set.

Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.

### 11.3.4 Falling trigger selection register (EXTI\_FTISR)

Address offset: 0x0C  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TR19	Res.	TR17	TR16											
												r/w		r/w	r/w
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
r/w															

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TR19**: Falling trigger event configuration bit of line 19  
 0: Rising trigger disabled (for Event and Interrupt) for input line  
 1: Rising trigger enabled (for Event and Interrupt) for input line.

Bits 18 Reserved, must be kept at reset value.

Bits 17:0 **TRx**: Falling trigger event configuration bit of line x (x = 17 to 0)  
 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 falling edge on an external interrupt line occurs during a write operation to the EXTI\_FTSR register, the pending bit is not set.*

*Rising and falling edge triggers can be set for the same interrupt line. In this case, both generate a trigger condition.*

### 11.3.5 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER 19	Res.	SWIER 17	SWIER 16
												rw		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 31:20 Reserved, must be kept at reset value.

Bits 19 **SWIER19**: Software interrupt on line 19

If the interrupt is enabled on this line in the EXTI\_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI\_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI\_PR (by writing a 1 into the bit).

Bits 18 Reserved, must be kept at reset value.

Bits 17:0 **SWIERx**: Software interrupt on line x (x = 17 to 0)

If the interrupt is enabled on this line in the EXTI\_IMR, writing a '1' to this bit when it is at '0' sets the corresponding pending bit in EXTI\_PR resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit of EXTI\_PR (by writing a 1 into the bit).

### 11.3.6 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
Res.	PR19	Res.	PR17	PR16											
												rc_w1		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															



Bits 31:20 Reserved, must be kept at reset value.

Bits 19 **PR19**: Pending bit on line 19

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.

Bits 18 Reserved, must be kept at reset value.

Bits 19:0 **PRx**: Pending bit on line x (x = 17 to 0)

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.

### 11.3.7 EXTI register map

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

**Table 29. Extended 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	Res.	Res.	Res.	MR[28:0]																												
	Reset value				1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	EXTI_EMR	Res.	Res.	Res.	MR[28:0]																												
	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
0x08	EXTI_RTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR19	Res.	TR[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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR19	Res.	TR[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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWIER19	Res.	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR19	Res.	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 [Section 2.2.2 on page 39](#) for the register boundary addresses.

## 12 Analog-to-digital converter (ADC)

### 12.1 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 three 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 [Table 10](#)).

### 12.2 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:
  - 1  $\mu$ s at 56 MHz (1.17  $\mu$ s at 72 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

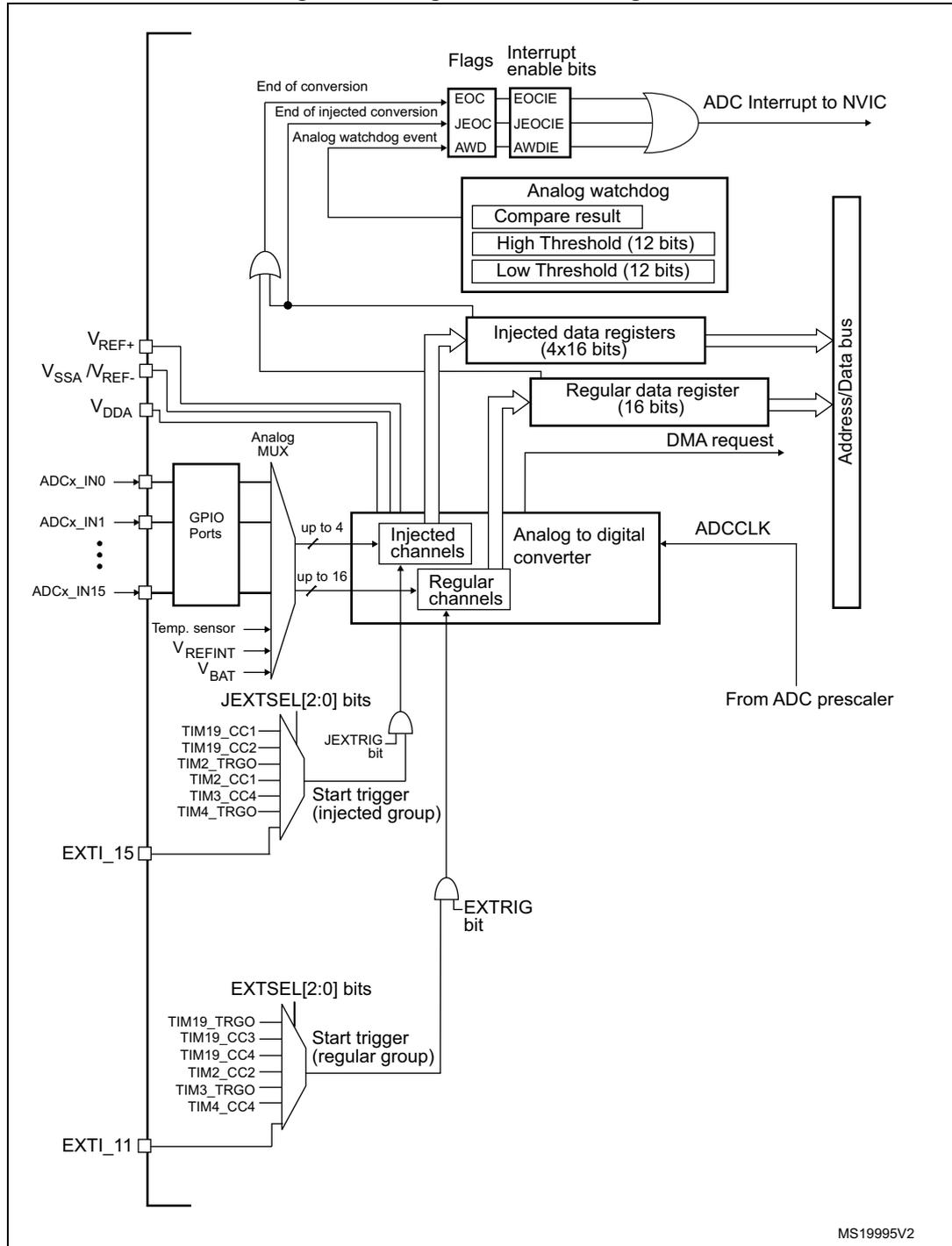
The block diagram of the ADC is shown in [Figure 25](#).

*Note:*  $V_{REF-}$ , if available (depending on package), must be tied to  $V_{SSA}$ .

### 12.3 ADC functional description

Figure 25 shows a single ADC block diagram and Table 30 gives the ADC pin description.

Figure 25. Single ADC block diagram



MS19995V2

Table 30. ADC pins

Name	Signal type	Remarks
$V_{REF+}$	Input, analog reference positive	The higher/positive reference voltage for the ADC, $2.4\text{ V} \leq V_{REF+} \leq V_{DDA}$
$V_{DDA}^{(1)}$	Input, analog supply	Analog power supply equal to $V_{DD}$ and $2.4\text{ V} \leq V_{DDA} \leq 3.6\text{ V}$
$V_{REF-}$	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 $V_{SS}$
ADC_IN[15:0]	Analog signals	Up to 16 external analog channels plus three internal: $V_{BAT}$ , temperature sensor, and $V_{REFINT}^{(2)}$

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.

### 12.3.1 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  $\mu\text{A}$ ).

### 12.3.2 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.

### 12.3.3 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/ $V_{REFINT}$ / $V_{BAT}$ internal channels

The Temperature sensor is connected to channel ADC\_IN16, the internal reference voltage  $V_{REFINT}$  is connected to ADC\_IN17, and the  $V_{BAT}$  is connected to ADC\_IN18. These two internal channels can be selected and converted as injected or regular channels.

#### 12.3.4 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 JEOP (End Of Conversion Injected) flag is set
  - and an interrupt is generated if the JEOPIE bit is set.

The ADC is then stopped.

#### 12.3.5 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.

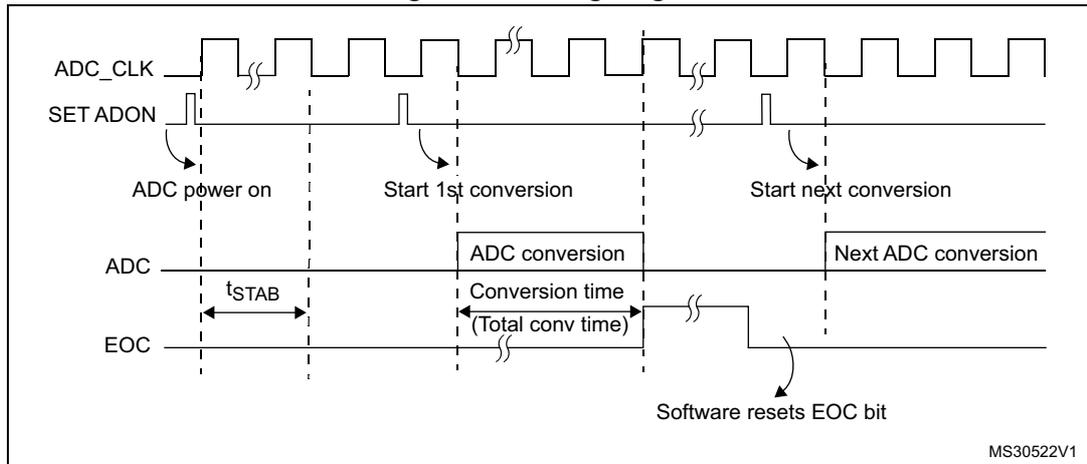
After each conversion:

- 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 JEOP (End Of Conversion Injected) flag is set
  - An interrupt is generated if the JEOPIE bit is set.

#### 12.3.6 Timing diagram

As shown in [Figure 26](#), 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 26. Timing diagram



### 12.3.7 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 12.5](#)).

The analog watchdog can be enabled on one or more channels by configuring the ADC\_CR1 register as shown in [Table 31](#).

Figure 27. Analog watchdog guarded area

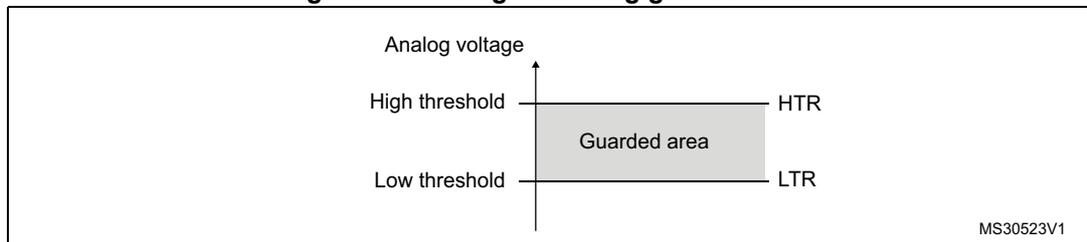


Table 31. Analog watchdog channel selection

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 <sup>(1)</sup> injected channel	1	0	1
Single <sup>(1)</sup> regular channel	1	1	0
Single <sup>(1)</sup> regular or injected channel	1	1	1

1. Selected by AWDCH[4:0] bits

### 12.3.8 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.

When using scan mode, DMA bit must be set and the direct memory access controller is used to transfer the converted data of regular group channels to SRAM after each update of the ADC\_DR register.

The injected channel converted data is always stored in the ADC\_JDRx registers.

### 12.3.9 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 28](#) 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.

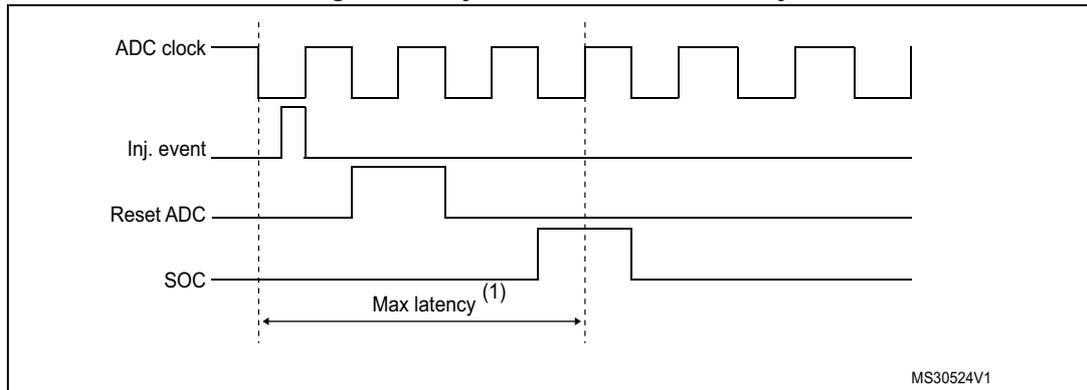
In this mode, external trigger on injected channels must be disabled.

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 28. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32F101xx and STM32F103xx datasheets.

### 12.3.10 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; an EOC event is generated at each conversion
- 2nd trigger: sequence converted 3, 6, 7; an EOC event is generated at each conversion
- 3rd trigger: sequence converted 9, 10; an EOC event is generated at each conversion
- 4th trigger: sequence converted 0, 1, 2; an EOC event is generated at each conversion

*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:* 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.

*It is not possible to use both auto-injected and discontinuous modes simultaneously. The user must avoid setting discontinuous mode for both regular and injected groups together. Discontinuous mode must be enabled only for one group conversion.*

## 12.4 Calibration

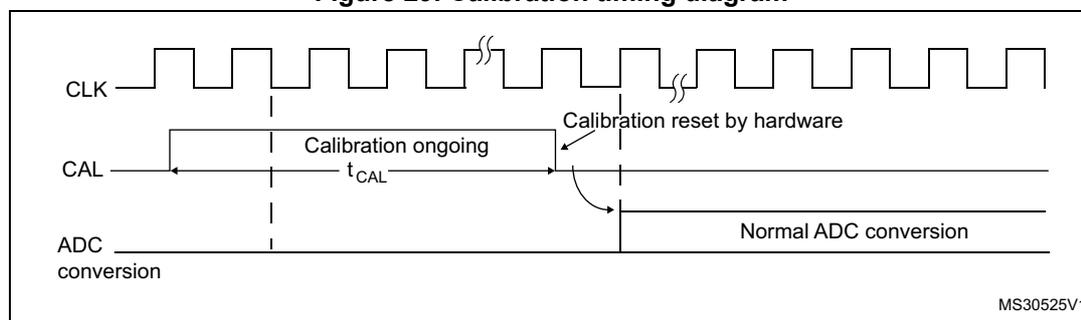
The ADC has an built-in self calibration mode. Calibration significantly reduces accuracy errors due to internal capacitor bank variations. During calibration, an error-correction code (digital word) is calculated for each capacitor, and during all subsequent conversions, the error contribution of each 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:* It is recommended to perform a calibration after each power-up.

*Before starting a calibration the ADC must have been in power-off state (ADON bit = '0') for at least two ADC clock cycles.*

Figure 29. Calibration timing diagram



## 12.5 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 30](#). and [Figure 31](#).

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 30. Right alignment of data

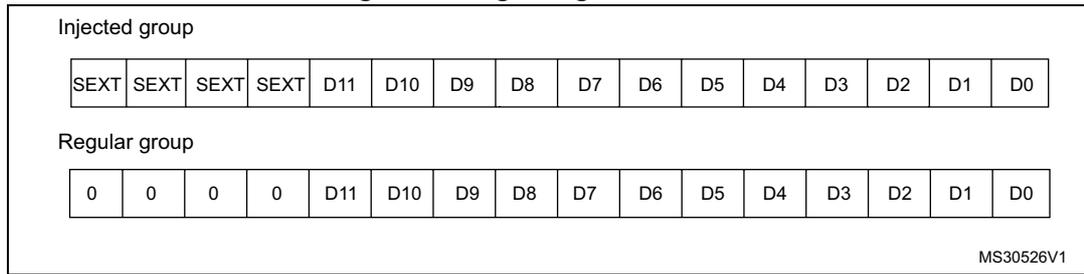
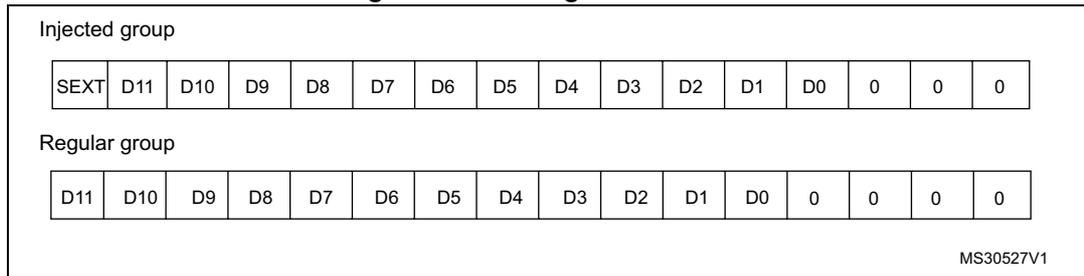


Figure 31. Left alignment of data



## 12.6 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:

$$T_{conv} = \text{Sampling time} + 12.5 \text{ cycles}$$

Example:

With an ADCCLK = 12 MHz and a sampling time of 1.5 cycles:

$$T_{conv} = 1.5 + 12.5 = 14 \text{ cycles} = \mu\text{s}$$

## 12.7 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 32. External trigger for regular channels for ADC1**

Source	Type	EXTSEL[2:0]
TIM19_TRGO	Internal signal from on-chip timers	000
TIM19_CC3		001
TIM19_CC4		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 33. External trigger for injected channels for ADC1**

Source	Connection type	JEXTSEL[2:0]
TIM19_CC1	Internal signal from on-chip timers	000
TIM19_CC2		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.

## 12.8 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.

## 12.9 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the ambient temperature ( $T_A$ ) of the device. The temperature sensor is internally connected to the ADC\_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  $\mu$ s.

The internal voltage reference ( $V_{REFINT}$ ) provides a stable (bandgap) voltage output for the ADC and Comparators.  $V_{REFINT}$  is internally connected to the ADC\_IN17 input channel. The precise voltage of  $V_{REFINT}$  is individually measured for each part by ST during production test and stored in the system memory area. It is accessible in read-only mode.

Figure 32 shows the block diagram of the temperature sensor.

When not in use, this sensor can be put in power down mode.

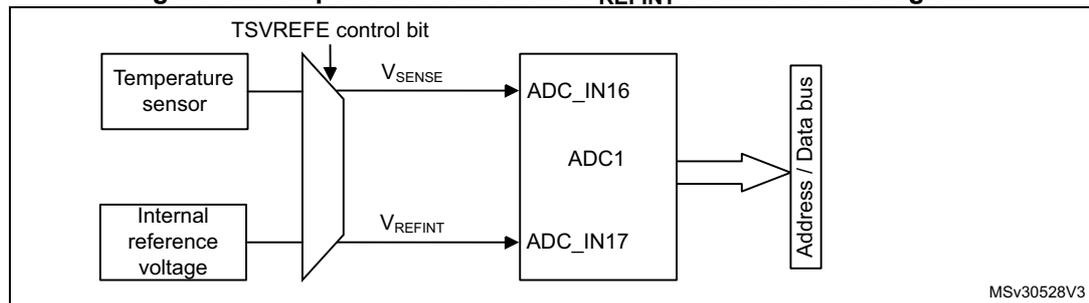
*Note:* The TSVREFE bit must be set to enable both internal channels: ADC\_IN16 (temperature sensor) and ADC\_IN17 ( $V_{REFINT}$ ) conversion.

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 for applications that detect temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor part should be used.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the datasheet for additional information.

**Figure 32. Temperature sensor and  $V_{REFINT}$  channel block diagram**



**Reading the temperature**

To use the sensor:

1. Select the ADC\_IN16 input channel.
2. Select a sample time of 17.1  $\mu$ s
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:  

$$\text{Temperature (in } ^\circ\text{C)} = \{(V_{25} - V_{SENSE}) / \text{Avg\_Slope}\} + 25.$$

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  $\mu$ V/°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.

## 12.10 Battery voltage monitoring

The VBAT\_MON bit in the SYSCFG\_CFGR1 register allows the backup battery voltage on the VBAT pin to be measured.

As the  $V_{BAT}$  voltage can be higher than  $V_{DDA}$ , to ensure the correct operation of the ADC, the  $V_{BAT}$  pin is internally connected to a bridge divider by 2. This bridge is automatically enabled when VBAT\_MON is set, to connect  $V_{BAT}/2$  to the ADC1\_IN18 input channel. Consequently, the converted digital value is half the  $V_{BAT}$  voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

## 12.11 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 34. 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

## 12.12 ADC registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

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

### 12.12.1 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
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.											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

### 12.12.2 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	
Res.								AWDEN	JAWDEN	Res.						
								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	JEOCIE	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.  
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  
10010: ADC analog Channel18  
Other values reserved.  
*Note: ADC1 analog Channel16, Channel17 and Channel 18 are internally connected to the temperature sensor, to  $V_{REFINT}$  and to  $V_{BAT}/2$  respectively.*

### 12.12.3 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			
Res.																								TSVRE FE				SWSTA RT				JSWST ART				EXTTR IG				EXTSEL[2:0]								Res.															
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			
JEXTT RIG				JEXTSEL[2:0]								ALIGN				Res.				DMA				Res.								RST CAL				CAL				CONT				ADON																			
rw				rw				rw				rw				rw				Res.				rw				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 19 TRGO event  
 001: Timer 19 CC3 event  
 010: Timer 19 CC4 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 19 CC1 event  
001: Timer 19 CC2 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 30](#) and [Figure 31](#).  
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.*



### 12.12.5 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
Res.		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
Res.		rw	rw	rw	rw	rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[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

*ADC1 analog Channel16 and Channel 17 are internally connected to the temperature sensor and to VREFINT, respectively.*

### 12.12.6 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	
Res.																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.				JOFFSETx[11:0]												
Res.				rw	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.

### 12.12.7 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
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				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.

### 12.12.8 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
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				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.

**12.12.9 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
Res.								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

**12.12.10 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
Res.		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

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

Bits 29:26 **SQ12[4:0]**: 12th 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

**12.12.11 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
Res.		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
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]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

### 12.12.12 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
	Res.										JL[1:0]		JSQ4[4:1]			
	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)<sup>(1)</sup>

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)

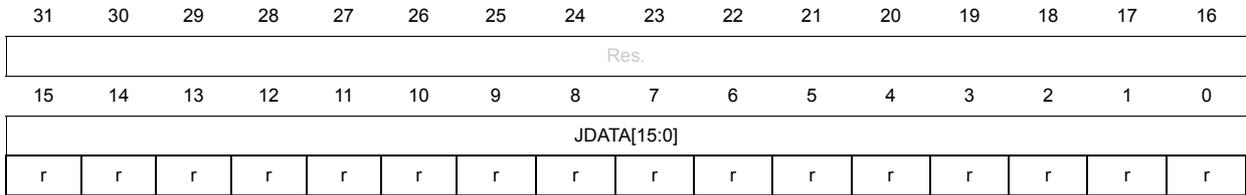
Bits 4:0 **JSQ1[4:0]**: 1st conversion in injected sequence (when JL[1:0] = 3)

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

### 12.12.13 ADC injected data register x (ADC\_JDRx) (x= 1..4)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000



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

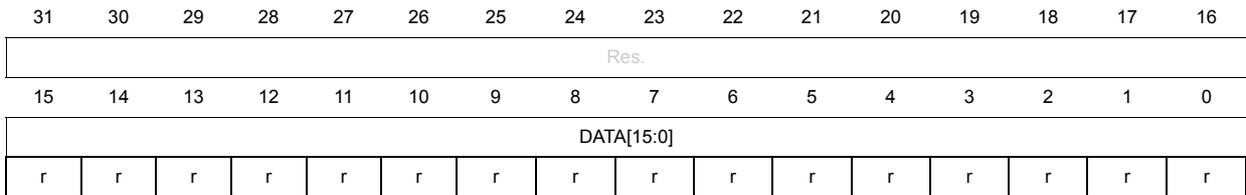
Bits 15:0 **JDATA[15:0]**: Injected data

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 30](#) and [Figure 31](#).

### 12.12.14 ADC regular data register (ADC\_DR)

Address offset: 0x4C

Reset value: 0x0000 0000



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 30](#) and [Figure 31](#).

### 12.12.15 ADC register map

The following table summarizes the ADC registers.

**Table 35. 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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT	JSTRT	JEOC	EOC	AWD				
	Reset value																												0	0	0	0	0				
0x04	ADC_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDEN	JAWDEN	Res.	Res.	Res.	Res.	Res.	Res.	DISC NUM [2:0]	Res.	Res.	JDISEN	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL [2:0]			Res.	JEXTTRIG	JEXTSEL [2:0]		ALIGN	Res.	Res.	DMA	Res.	Res.	Res.	Res.	Res.	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET1[11:0]														
	Reset value																						0	0	0	0	0	0	0	0	0	0	0				
0x18	ADC_JOFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET2[11:0]														
	Reset value																						0	0	0	0	0	0	0	0	0	0	0				
0x1C	ADC_JOFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET3[11:0]														
	Reset value																						0	0	0	0	0	0	0	0	0	0	0				
0x20	ADC_JOFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JOFFSET4[11:0]														
	Reset value																						0	0	0	0	0	0	0	0	0	0	0				
0x24	ADC_HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT[11:0]														
	Reset value																						0	0	0	0	0	0	0	0	0	0	0				
0x28	ADC_LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT[11:0]														
	Reset value																						0	0	0	0	0	0	0	0	0	0	0				
0x2C	ADC_SQR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L[3:0]			SQ16[4:0]				SQ15[4:0]				SQ14[4:0]				SQ13[4:0]												
	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				

Table 35. 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
0x30	ADC_SQR2	Res.	Res.	SQ12[4:0]				SQ11[4:0]				SQ10[4:0]				SQ9[4:0]				SQ8[4:0]				SQ7[4:0]									
	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
0x34	ADC_SQR3	Res.	Res.	SQ6[4:0]				SQ5[4:0]				SQ4[4:0]				SQ3[4:0]				SQ2[4:0]				SQ1[4:0]									
	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
0x38	ADC_JSQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JSQ4[4:0]				JSQ3[4:0]				JSQ2[4:0]				JSQ1[4:0]							
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x3C	ADC_JDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	ADC_JDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	ADC_JDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	ADC_JDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	ADC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Regular DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2 on page 39](#) for the register boundary addresses.

## 13 Sigma-delta analog-to-digital converter (SDADC)

### 13.1 Introduction

The SDADC module is a high-performance and low-power sigma-delta analog-to-digital converter, featuring 16-bit resolution and 9 differential analog channels with selectable gains.

The conversion speed is up to 16.6 ksps (kilo-samples per second) for each SDADC when converting multiple channels and up to 50 ksps per SDADC if only one channel conversion is used. There are two conversion modes: single conversion mode and continuous mode, capable of automatically scanning any number of channels. The data can be automatically stored in a system RAM buffer, reducing the software overhead.

A flexible timer triggering system can be used to control the start of conversion of the three SDADCs. This timing control is capable of triggering simultaneous conversions or inserting a programmable delay between the SDADCs.

Reference voltage for SDADC can be selected from external reference pins, internal 1.2/1.8V reference or SDADC analog power supply.

Four power modes are supported: Normal, Slow, Standby and Power down. In Standby mode, references stay powered on to reduce the startup time.

## 13.2 SDADC main features

- 16-bit sigma-delta architecture
- 5 differential input pairs, or 9 single-ended inputs, or a combination
- High-performance data throughput:
  - 16.6 ksp/s input sampling rate when multiplexing between different channels
  - 50 ksp/s input sampling rate for single-channel operation
- Self-calibration
- 7 gain settings from 0.5x to 32x
- Full-scale single-ended conversion mode referenced to ground in addition to differential mode
- Each channel can choose between 3 user-defined configurations, where each configuration specifies:
  - Analog gain at the input of the channel
  - Conversion mode: differential vs. full-scale single-ended referenced to ground
  - Calibration offset
- Selectable reference voltage which determines the input signal range and digital end-of-scale:
  - Internal VDDSDx:  $V_{REF} = \text{analog supply}$ , 2.4 V-3.6 V (2.2 V-3.6 V in Slow mode)
  - Internal bandgap:  $V_{REF} = 1.2 \text{ V}$
  - Internal bandgap:  $V_{REF} = 1.8 \text{ V}$
  - External:  $V_{REF} = 1.1 \text{ V}$  to analog supply
- Continuous conversion
- Start-of-conversion synchronization with
  - Software trigger
  - Internal timers
  - External events
  - start-of-conversion of another (first) SDADC
- “Regular” conversions can be requested at any time or even in continuous mode without having any impact on the timing of “injected” conversions
- Two’s-complement output format
- DMA may be used to read the conversion data
- End of conversion, overrun, and end of calibration interrupts
- 3 low-power modes (refer to the datasheet for the current consumption values):
  - Slow mode where the device operates from a reduced clock frequency
  - Standby mode
  - Power down mode
- All three SDADCs can share up to 23 input pins which may be configured in any combination of single-ended (up to 21) or differential inputs (up to 11).

### 13.3 SDADC pins

Table 36. ADC pins

Name	Signal Type	Remarks
VREFSD+	Input or In/Out, positive analog reference	When the external reference is selected (REFV=00), this pin must be driven externally to a voltage between 1.1 V and VDDSDx. When an internal reference is selected (REFV is 01, 10, or 11), this pin must have an external capacitance connected to VREFSD-.
VREFSD-	Input, negative analog reference	This pin, when present, must be driven to the same voltage level as VSSSD.
VDDSDx	Input, analog supply	Analog power supply. Must be greater than 2.4 V (or 2.2 V in Slow mode) and less than 3.6 V.
VSSSD	Input, analog supply ground	Analog ground power supply.
SDADCx_AIN[8:0]P	Analog input	Positive differential analog inputs for the 9 channels.
SDADCx_AIN[8:0]M	Analog input	Negative differential analog inputs for the 9 channels.

### 13.4 SDADC clock

The clock source for SDADC is derived from the system clock. This clock is divided by a selectable divider with 50% duty cycle.

Any of the following division ratios can be selected:

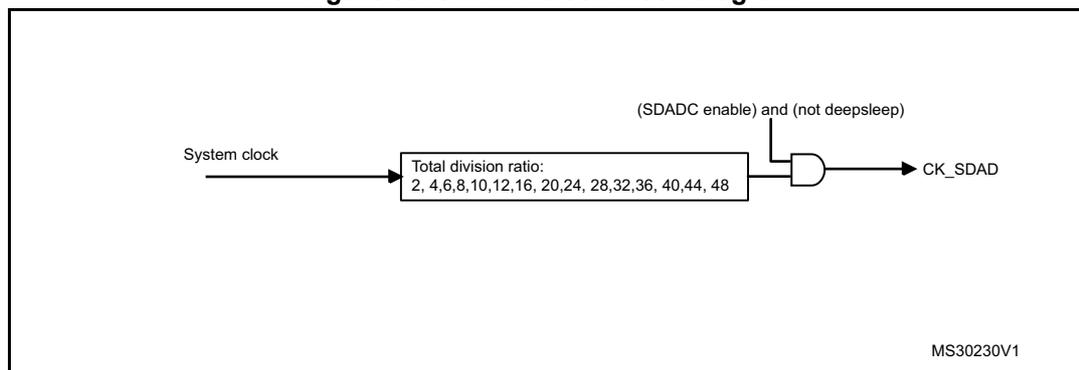
- 2, 4, 6, 8, 10, 12, 14, 16, 20, 24, 28, 32, 36, 40, 44, 48.

The SDADC clock is automatically stopped in deepsleep mode.

The maximum operating frequency of the SDAC is 6 MHz. Its minimum operating frequency is 500 kHz.

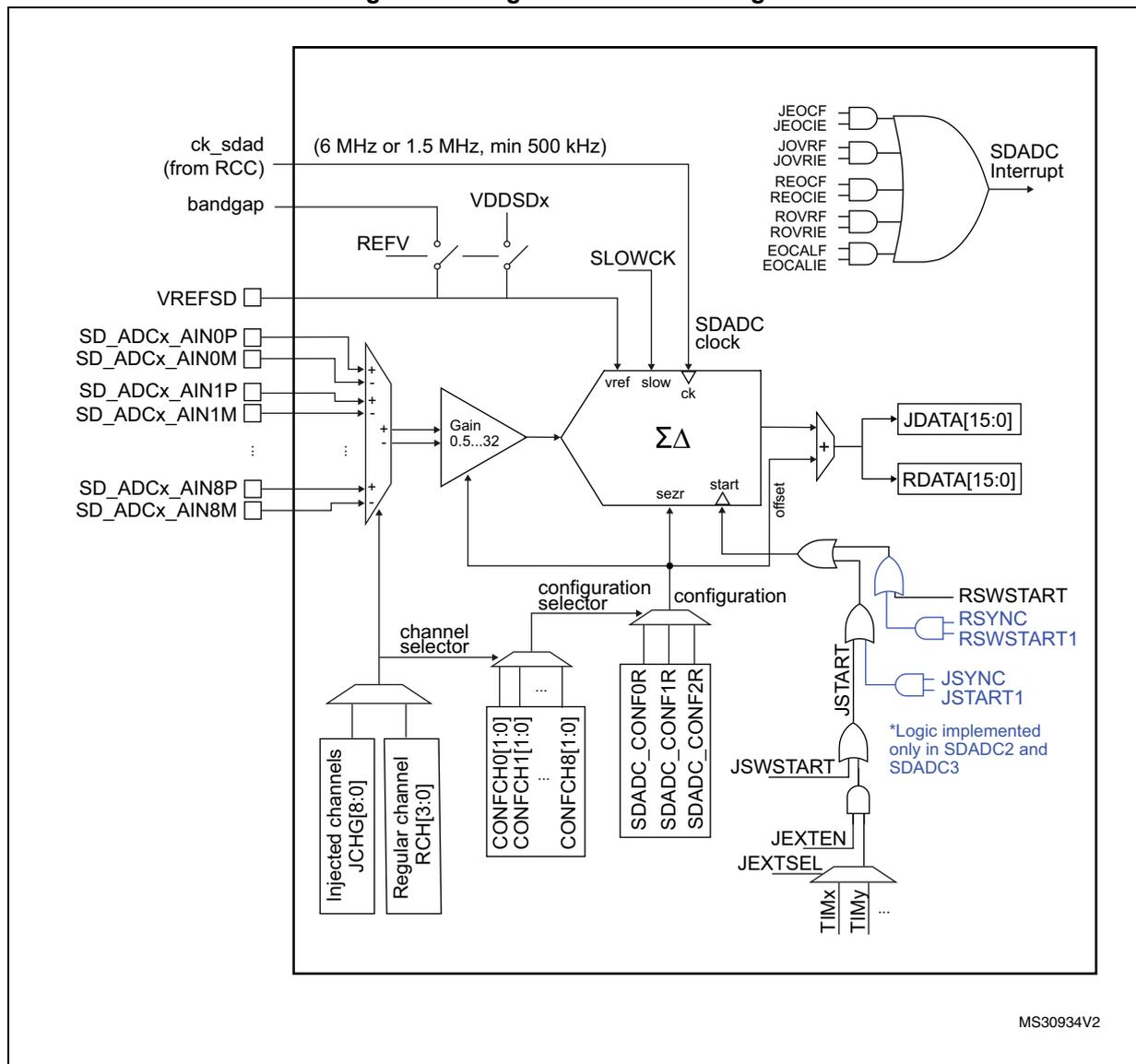
A detailed diagram of SDADC clock is given in [Section 7.2.9: SDADC clock on page 103](#)

Figure 33. SDADC clock block diagram



### 13.5 SDADC functional description

Figure 34. Single SDADC block diagram



MS30934V2

#### 13.5.1 SDADC on-off control

The SDADC is enabled by setting the ADON bit in the SDADC\_CR2 register. After the SDADC is powered on, it needs 100  $\mu$ s to stabilize before it can start a conversion or launch calibration (unless PDI=1, see next section). An action requested in the meantime will be automatically started as soon as stabilization is complete. The end of stabilization is signalled by bit STABIP in the SDAC\_ISR register.

Clearing ADON stops any conversion which may be in progress and puts the SDADC in power down mode.

### 13.5.2 Power down and Standby low-power modes

In order to reduce consumption, the SDADC can be automatically put into either Standby mode or power down mode when it is idle. "Idle" is defined as when RCIP=0, JCIP=0, and CALIBIP=0.

Setting PDI in the SDADC\_CR1 register causes the SDADC to enter power down mode when idle, where it consumes about 10  $\mu$ A instead of up to 1.2 mA (see datasheet for exact values). Whenever exiting power down mode, a period of 100  $\mu$ s is needed for stabilization. During stabilization, a conversion may be requested, but it will not start until stabilization is complete (STABIP=0).

Similarly, setting SBI in the SDADC\_CR1 register puts the SDADC in Standby mode when idle, where it consumes a maximum of about 200  $\mu$ A. Whenever exiting Standby mode conversions, a period of 50  $\mu$ s is required for stabilization.

While the SDADC is stabilizing, the stabilization in progress status bit, STABIP, in SDADC\_ISR is set to '1'.

When enforcing the stabilization times, the SDADC measures these durations assuming that the prescaler outputs a clock at 6 MHz. This means that the 100  $\mu$ s period after power on is defined as 600 SDADC clock cycles (or 150 if SLOWCK=1, where the SDADC frequency should be 1.5 MHz), and the 50  $\mu$ s period after exiting Standby mode is defined as 300 SDADC clock cycles (or 75 if SLOWCK=1).

When SBI=1, if a conversion or calibration is requested within the first 50  $\mu$ s after ADON is activated, a stabilization period of 100  $\mu$ s (rather than 50  $\mu$ s) is observed starting from the moment that the conversion or calibration is requested.

### 13.5.3 SDADC clock

The SDADC clock, which is used to drive the analog logic, is generated by the RCC block. When not in Slow mode, its prescaler should be configured so that the SDADC can run at its maximum frequency of 6 MHz. The minimum operating frequency of the SDADC is 500 kHz.

#### Slow mode

Setting the SLOWCK bit in the SDADC\_CR1 register puts the SDADC in Slow mode. When SLOWCK=1, the analog consumes less and is able to operate down to a voltage of 2.2 V, but the frequency of the SDADC clock must be reduced to 1.5 MHz. The minimum frequency is still 500 kHz in Slow mode.

### 13.5.4 Channel selection

There are 9 multiplexed channels which can be selected for conversion using the injected channel group and using the regular channel.

The **injected channel group** is a selection of any or all of the 9 channels. JCHG[8:0] in the SDADC\_JCHGR register selects the channels of the injected group, where JCHG[i]=1 means that channel i is selected.

Injected conversions are always executed in scan mode, which means that each of the selected channels are converted in series. The highest channel (channel 8, if selected) is converted first, followed immediately by the next lower channel until all the channels selected by JCHG[8:0] have been converted.

Injected conversions can be launched by software or by a trigger. They are never interrupted by regular conversions.

The **regular channel** is a selection of just one of the 9 channels. RCH[3:0] in the SDADC\_CR2 register indicates the selected channel.

Regular conversions can be launched only by software (not by a trigger). A sequence of continuous regular conversions is temporarily interrupted when a injected conversion is requested.

### 13.5.5 Differential and single-ended modes

Each SDADC channel has 2 differential inputs (positive and negative: SDADCx\_AIN[n]P, SDADCx\_AIN[n]M). Configuring of those inputs to pins connection can be obtained several measurement modes.

**Differential mode:** Simple mode where both - positive and negative inputs - are connected to external pins. The output signal is positive or negative depending on the connected signal polarity. The corresponding SE[1:0] bits must be set to "00" (see [Section 13.5.6: Configuring the analog inputs](#)) to select this mode.

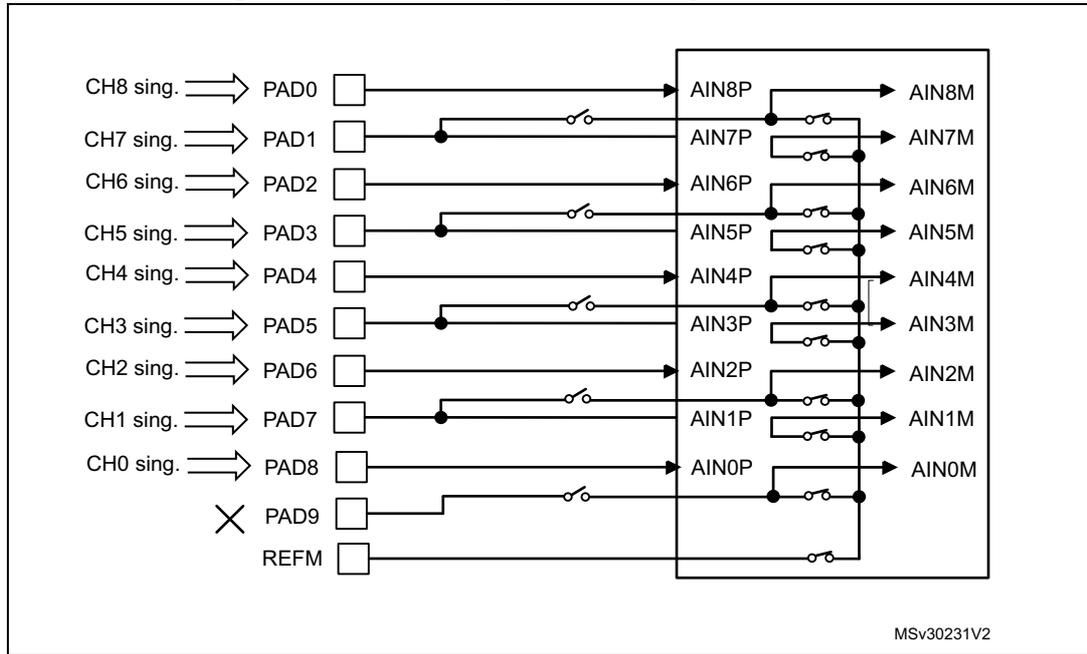
In addition to this differential mode, conversions may be performed in one of two single-ended modes. When in single-ended mode, the negative input is set to 0 V internally, leaving the corresponding pin for the negative input (SDADCx\_AIN[n]M) free to be used for other purposes. The signal to be measured is applied to the positive input.

**Single-ended offset mode:** The corresponding SE[1:0] bits must be set to "01" (see [Section 13.5.6: Configuring the analog inputs](#)) to select this mode. The output signal is always positive, thus excluding the negative half of the dynamic range. In this mode, the signal to noise ratio (SNR) is degraded by 6 dB.

**Single-ended zero reference mode:** The corresponding SE[1:0] bits must be set to "11" (see [Section 13.5.6: Configuring the analog inputs](#)) to select this mode. This mode injects an offset of half scale to the SDADC thus maintaining the full positive/negative dynamic range like in differential mode. In this mode, the offset is dependent on gain variations.

Examples of possible modes

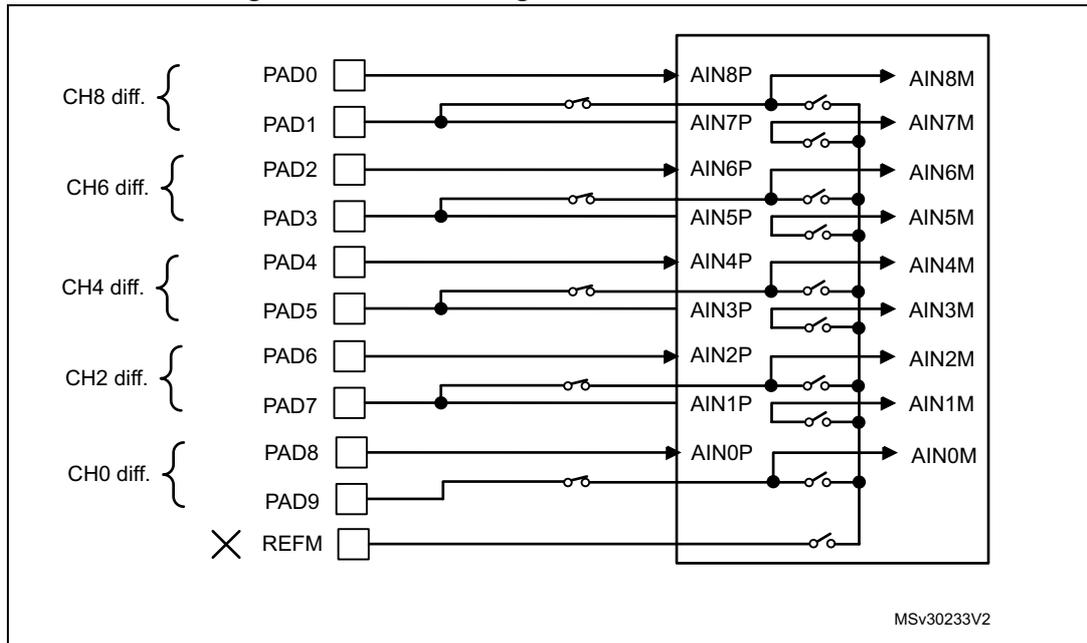
Figure 35. Switch configuration in single-ended mode



- All analog switches on the left are open.
- All analog switches on the right are closed.
- CH8 to CH0 are used in single-ended mode.
- REFM is used.
- PAD 9 is not used.

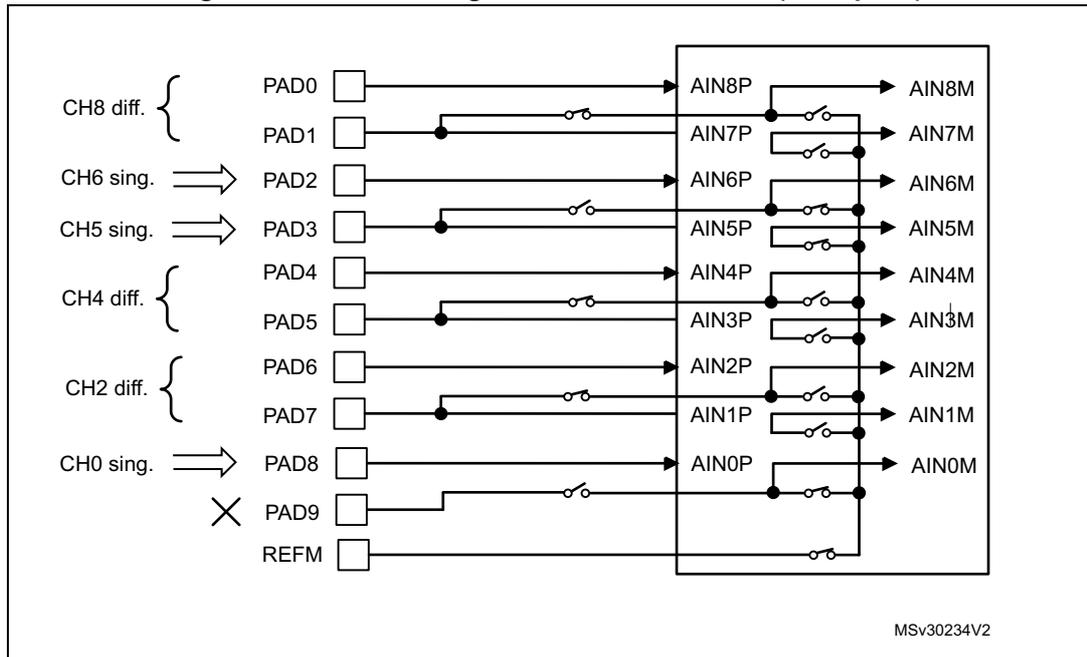
MSv30231V2

Figure 36. Switch configuration in differential mode



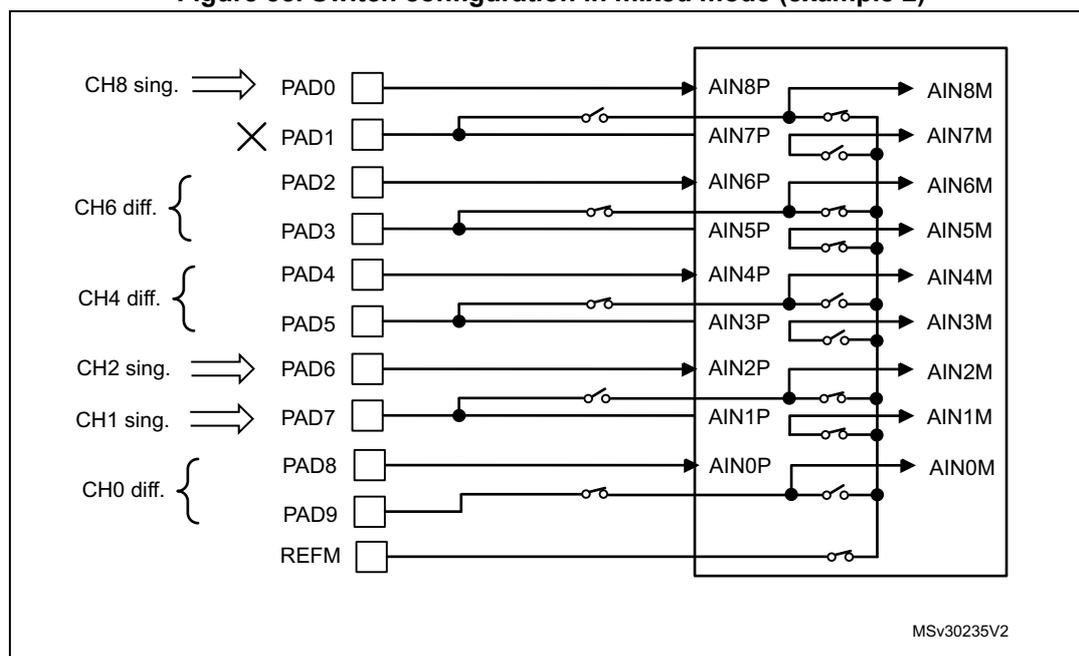
- All analog switches on the left are closed.
- All analog switches on the right are open.
- CH8, CH6, CH4, CH2, and CH0 are in differential mode.
- CH7, CH5, CH3 and CH1 are not used.
- REFM is not used.

Figure 37. Switch configuration in mixed mode (example 1)



- CH8, CH4 and CH2 are used as differential.
- CH6, CH5 and CH0 are used in single-ended mode.
- REFM is used.
- PAD 9 is not used.

Figure 38. Switch configuration in mixed mode (example 2)



- CH6, CH4 and CH0 are used as differential.
- CH8, CH2 and CH1 are used in single-ended mode.
- REFM is used.
- PAD 1 is not used.

### 13.5.6 Configuring the analog inputs

The following parameters must be configured for the analog inputs:

- **Gain.** There are 5 analog gain settings (1/2x, 1x, 2x, 4x, 8x) and 2 digital gain settings (16x, 32x).
- **Single-ended modes (SE)** (see [Section 13.5.5: Differential and single-ended modes](#)).
- **Common mode.** The common mode setting (VSSSD, VDDSDx, or VDDSDx/2) is used only in the determination of the offset during the calibration sequence.
- **Offset.** The 12-bit offset is applied to the conversion results. This value varies from part to part and also depends on the common mode, on the SE setting, and on the gain if SE=11. This value can be determined automatically during the calibration sequence or it can be written by software.

Three distinct configurations can be specified using the SDADC\_CONF0R, SDADC\_CONF1R, and SDADC\_CONF2R registers. Each SDADC\_CONFxR register contains the fields GAINx[2:0], COMMONx[1:0], SE[1:0], and OFFSETx[11:0].

Each individual input can select one of the three configurations, by way of the SDADC\_CONFCHR1 and SDADC\_CONFCHR2 registers.

### 13.5.7 Launching calibration and determining the offset values

Calibration can be used to determine the offset values of the configurations defined in each of the three SDADC\_CONFxR registers. OFFSETx[11:0] is determined based on GAINx[2:0], COMMONx[1:0], and SE[1:0] (where x is 0, 1, or 2).

During the offset calibration positive and negative SDADC inputs are shorted internally and connected to common voltage given by COMMONx[1:0] setting. GAINx[2:0] is applied and then is performed a conversion which determines the OFFSETx[11:0] value (12-bit signed value).

The calibration sequence consists of the following steps:

- The SDADC\_CONFxR registers must be written. If only 1 distinct configuration is used, then it must be defined in the SDADC\_CONF0R register. If only 2 distinct configurations are used, then they must be defined in the SDADC\_CONF0R and SDADC\_CONF1R registers.
- The SDADC\_CR2 register is written:
  - “00” written to CALIBCNT[1:0] if only OFFSET0 is to be determined, “01” for both OFFSET0 and OFFSET1, or “10” to determine all three offset values,
  - ‘1’ must be written to STARTCALIB to launch calibration.
- The calibration sequence then executes, taking 30720 ADC cycles (5.12 ms at 6 MHz) for each offset calculation. Thus, all three offsets are calculated, the entire sequence lasts 15.36 ms at 6 MHz.
- The offset values are automatically stored in the corresponding OFFSETx[11:0] fields.

### 13.5.8 Launching conversions

Injected conversions can be launched using the following methods:

- Software: writing ‘1’ to the JSWSTART bit in the SDADC\_CR2 register.
- Trigger: the JEXTSEL[2:0] bits select the trigger signal while the JEXTEN bit activates the trigger and selects the active edge at the same time.
- Synchronous with SDADC1: for SDADC2 or SDADC3, an injected conversion is automatically launched when an action in SDADC1 causes its own injected conversion sequence to start. Each injected conversion in SDADC2 or SDADC3 is always executed according its local configuration settings (JDS, JCONT, JCHG, etc.).

Each time an injected conversion is launched, all of the selected channels in the injected group are converted sequentially, starting with the lowest channel (channel 0, if selected).

Only one injected conversion can be pending or ongoing at a given time. Thus, any request to launch an injected conversion is ignored if another request for an injected conversion has already been issued but not yet completed.

Regular conversions can be launched using the following methods:

- Software: by writing ‘1’ to RSWSTART in the SDADC\_CR2 register.
- Synchronous with SDADC1: for SDADC2 or SDADC3, a regular conversion is automatically launched when an action in SDADC1 causes its own regular conversion sequence to start. Each regular conversion in SDADC2 or SDADC3 is always executed according its local configuration settings (RCONT, RCH, etc.).

Only one regular conversion can be pending or ongoing at a given time. Thus, any request to launch a regular conversion is ignored if another request for an regular conversion has already been issued but not yet completed.

### 13.5.9 Continuous and fast continuous modes

Setting the JCONT bit in the SDADC\_CR2 register causes injected conversions launched by software to execute in continuous mode. If software writes ‘1’ to the JCONT bit at the

same time as it writes '1' to the JSWSTART bit, the same scan sequence is performed repeatedly, always starting over at the highest channel after the lowest channel is finished.

Similarly, setting the RCONT bit causes regular conversions to execute in continuous mode. RCONT=1 means that the channel selected by the RCH[3:0] bits is converted repeatedly after '1' is written to the RSWSTART bit.

The sequence of injected conversions executing in continuous mode can be stopped by writing '0' to the JCONT bit. After clearing the JCONT bit, only the on-going conversion will be completed; the scan sequence is interrupted, and thus the final conversion will not be the last (lowest) selected channel unless it was the one being converted when the JCONT bit was cleared.

Similarly, writing '0' to the RCONT bit stops continuous regular conversions, allowing only the currently executing conversion to complete.

If just a single channel is selected in continuous mode (either by executing a regular conversion or by executing an injected conversion with only one channel selected), the sampling rate can be increased three fold by setting the FAST bit in the SDADC\_CR2 register. The conversion of each channel normally requires 360 SDADC clock cycles (60  $\mu$ s at 6 MHz). In fast continuous mode (FAST=1), the first conversion takes still 360 SDADC clocks, but then each subsequent conversion finishes in 120 SDADC clocks.

### 13.5.10 Request precedence

In summary, the calibration sequence has the highest precedence, followed by injected conversions, while regular conversions have the lowest priority. However, an individual conversion which is already in progress is never interrupted by the request for another action. Also, a request is ignored if a like action is already pending or in progress. Finally, no action can start before stabilization has finished. The following text gives examples and more details.

Injected conversions can not be launched if another injected conversion is pending or already in progress: any request to launch an injected conversion (either by JSWSTART or by a trigger) is ignored when the bit JCIP (in the SDADC\_ISR register) is '1'.

Similarly, regular conversions can not be launched if another regular conversion is pending or already in progress: any request to launch a regular conversion (using RSWSTART) is ignored when the RCIP bit in the SDADC\_ISR register is '1'.

However, if an injected conversion is requested while a regular conversion is already in progress (or vice-versa), the injected conversion is launched as soon as the regular conversion is finished (or vice-versa, assuming that the injected scans sequence is finished and JCONT=0).

Injected conversions have precedence over regular conversions in that an injected conversion can temporarily interrupt a sequence of continuous regular conversions (after the current conversion finishes). When the sequence of injected conversions finishes (at the end of the scan sequence or by writing '0' to the JCONT bit in the case of continuous injected conversion mode), the continuous regular conversions start again if the RCONT bit is still set.

Precedence matters also when actions are initiated by the same write to SDADC, or if multiple actions are pending at the end of another action. For example, suppose that while stabilization is in process (STABIP=1), a single write operation to SDADC\_CR2 writes '1' to both the RSWSTART and STARTCALIB bits, requesting a regular conversion as well as a calibration sequence. Then a trigger event occurs which requests an injected conversion,

still during stabilization. When stabilization finishes, precedence dictates that the calibration sequence will execute first, followed by the injected scan sequence, and then finally the regular conversion is performed.

### 13.5.11 Launching conversions with deterministic timing

In applications where certain conversions must be launched at precise intervals, it is a problem if these conversions get delayed by another conversion which is already in progress. This issue can be resolved by setting the JDS (delay start of injected conversions) bit in the SDADC\_CR2 register.

When JDS=1, the start of each injected conversion is delayed by 500 cycles, during which time no new regular conversions may be launched. Since no conversion can take longer than 360 cycles once it is started, there is guaranteed to be no regular conversions which are in progress at the end of the delay. Note that if PDI=1 (power down mode when idle) and SLOWCK=0 (when the SDADC clock frequency can be as high as 6 MHz), the delay is increased from 500 cycles to 600 cycles since the SDADC needs that many cycles to stabilize as it wakes from power down mode.

In this manner, applications can launch regular conversions at any time without affecting the timing of the injected conversions. If continuous regular conversions are executing, they will restart automatically after the injected conversions are complete.

### 13.5.12 Reference voltage

The reference voltage, common to all three sigma-delta ADCs (SDADC1, SDADC2, and SDADC3), is always seen on the  $V_{REF}$  pin

- When REFV (SDADC\_CR1) is "00" (its default value), the  $V_{REF}$  pin must be driven externally to a voltage between 1.1V and  $V_{DDA}$ .
- When REFV=01, the  $V_{REF}$  pin is forced internally to the 1.2V bandgap voltage and must be connected externally to a capacitance coupled to  $V_{REFM}$  if present, or  $V_{SSA}$  otherwise.
- When REFV=10, the  $V_{REF}$  pin is forced internally to the 1.8V bandgap voltage and must be connected externally to a capacitance coupled to  $V_{REFM}$  if present, or  $V_{SSA}$  otherwise.
- When REFV=11, the  $V_{REF}$  pin is forced internally to  $V_{DDSDx}$ . It must be externally connected to a capacitance coupled to  $V_{REFSD-}$ .
  - If SDADC1 or SDADC2 are enabled through ENSD1 or ENSD2 bits in PWR\_CR register then  $V_{DDSD1/2}$  must be at the same voltage level as  $V_{DDSD3}$ .
  - If SDADC1 and SDADC2 are disabled through ENSD1 and ENSD2 bits in PWR\_CR register then  $V_{DDSD12}$  can be lower than  $V_{DDSD3}$ .

The REFV[1:0] control bits are available only in the register set of SDADC1.

For applications which do not use the SDADC, the  $V_{REFSD+}$  pin must not be left floating. The  $V_{REFSD+}$  pin must be tied to  $V_{DD}$ , or software must set REFV to "11". The  $V_{REFSD-}$  pin must always be grounded.

The selected reference voltage is always present on the  $V_{REFSD+}$  pin. This pin must be decoupled by a capacitor (1  $\mu$ F recommended). If  $V_{DDSDx}$  is selected through the reference voltage selection bits (REFV="11" in SDADC\_CR1 register), the application must first configure REFV and then wait for at least 2 ms before enabling the SDADC (ADON=1

in SDADC\_CR2 register). The 1  $\mu$ F decoupling capacitor must be fully charged before enabling the SDADC.

The voltage on VREFSD+ pin must meet the following conditions:

- It must always be less than VDDSD3 (in particular when no SDADC is used).
- it must be always less than VDDSD12 if SDADC1 or SDADC2 is enabled through ENSD1 or ENSD2 bits in PWR\_CR register.

### 13.5.13 Analog input signal ranges

The input analog voltage on input channel pins (SDADCx\_AIN[8:0]P, SDADCx\_AIN[8:0]M) must be in the SDADC power supply range (VSSSD, VDDSDx) for all selected measurement modes and gains.

The input analog voltage range corresponding to full-scale SDADC output data range depends on the measurement mode ([Section 13.5.5: Differential and single-ended modes](#)), on the selected channel gain, and on the selected reference voltage (configured through REFV[1:0] bits):

- In differential mode, the full-scale differential voltage,  $V_{IN}$ , ranges between SDADCx\_AIN[8:0]P and SDADCx\_AIN[8:0]M:

$$V_{IN} = \frac{-VREFSD}{2 \times \text{gain}} \text{ to } \frac{VREFSD}{2 \times \text{gain}}$$

- In single ended offset mode, the full-scale differential voltage,  $V_{IN}$ , ranges between SDADCx\_AIN[8:0]P and  $V_{SSA}$ :

$$V_{IN} = V_{SSA} \text{ to } \frac{VREFSD}{2 \times \text{gain}}$$

- In single ended zero reference mode, the full-scale differential voltage,  $V_{IN}$ , ranges between SDADCx\_AIN[8:0]P and  $V_{SSA}$ :

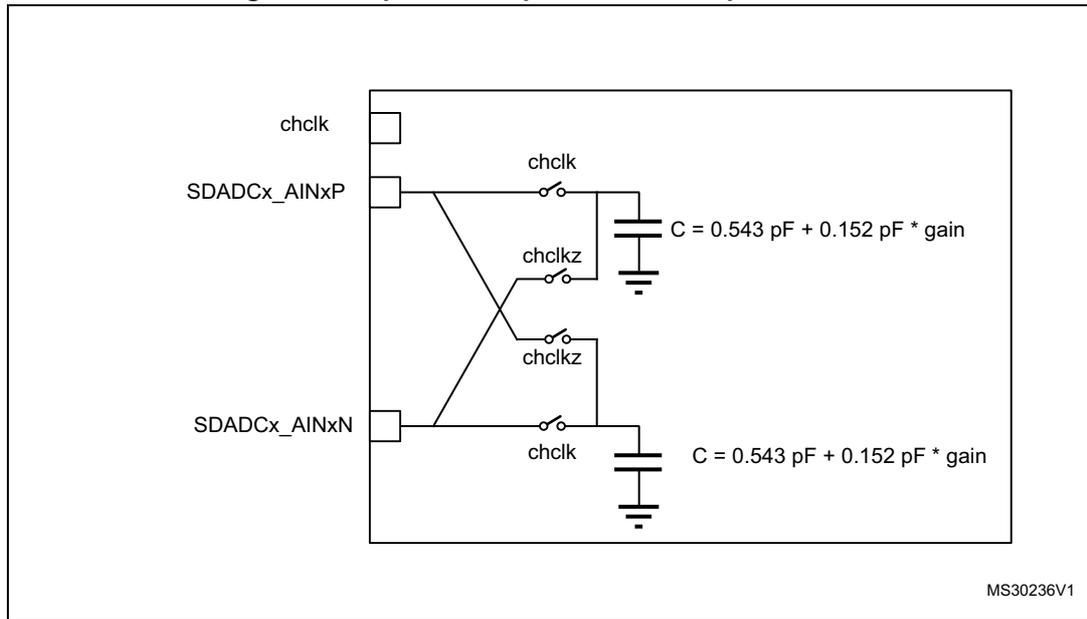
$$V_{IN} = V_{SSA} \text{ to } \frac{VREFSD}{\text{gain}}$$

where  $VREFSD = VREFSD+ - VREFSD-$ .

### 13.5.14 Input impedance of SDADC analog input and VREFSD reference voltage

Input impedance of SDADC depends from the selected SDADC clock, selected gain and if conversion is in progress. The input equivalent circuit is on the following figure.

Figure 39. Equivalent input circuit for input channel



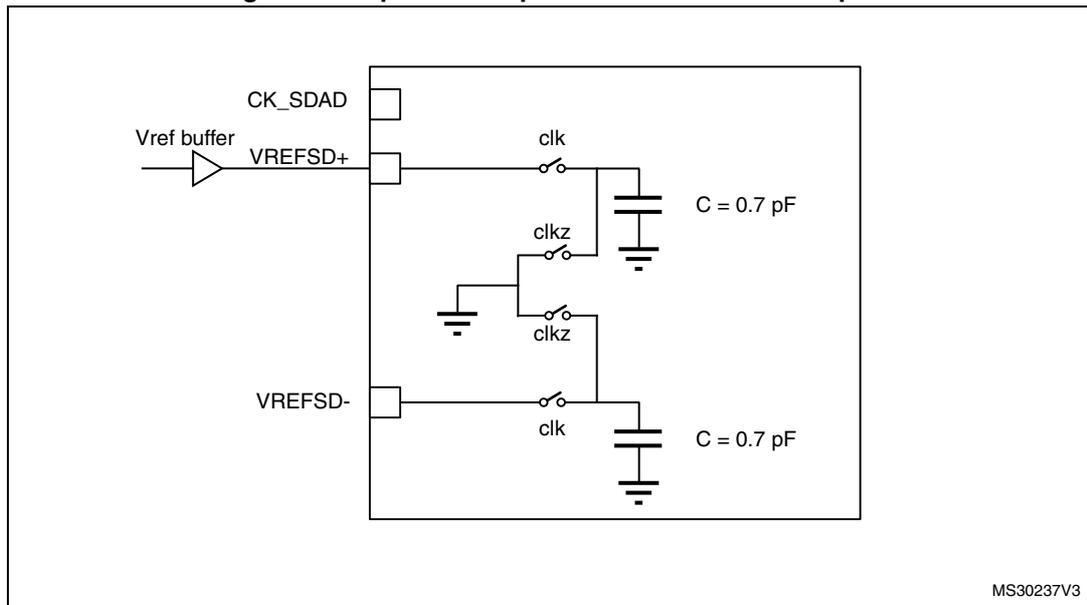
Note: Gain can be from 0.5x to 8x (16x and 32x are digital gains)  
 Both chclk and chclkz are 0 when the channel is not active (not sampled) and both switch with opposite phases when the channel is active.

The average impedance during the channel conversion is:

$$R_{in} = \frac{1}{2 \cdot F_{clk} \cdot C}$$

Input equivalent circuit for external reference voltage (VREFSD) input is shown in the next figure.

Figure 40. Equivalent input circuit for VREFSD input



The average impedance of external VREFSD+ input during SDADC operation is:

$$R_{in} = \frac{1}{F_{clk} \cdot C}$$

## 13.6 SDADC registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

### 13.6.1 Register write protection

	ADON=1	INITRDY=0 and ADON=1	JCIP=1	RCIP=1	CALIBIP=1
RSYNC		ro			
JSYNC		ro			
PDI/SBI	ro				
SLOWCK	ro				
REFV	ro <sup>(1)</sup>				
FAST		ro			
RSWSTART		ignored <sup>(2)</sup>		ignored	
JSWSTART		ignored <sup>(2)</sup>	ignored		
JEXTEN		ro			
JEXTSEL		ro			
JDS		ro			
STARTCALIB		ignored <sup>(2)</sup>			ignored
CALIBCNT		ro			
SDADC_JCHGR	rwnz	rwnz	rwnz	rwnz	rwnz
SDADC_CONFxR		ro			
SDADC_CONFCHR <sub>x</sub>		ro			

- REFV can be modified only when all of the SDADC modules are disabled (ADON=0 for all SDADCs).
- The "START" bits are ignored when INIT=1 (as soon as initialization mode is requested). All bits can be modified when ADON=0.

#### Table legend

ro = read only

rwnz = read and write-non-zero (writes of all-zero values are ignored)

blank = read and write (no protection)

### 13.6.2 SDADC control register 1 (SDADC\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INIT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDMAEN	JDMAEN
rw														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSYNC	JSYN C	Res.	PDI	SBI	SLOW CK	REFV[1:0]		Res.	Res.	Res.	ROVRI E	REOCIE	JOVRIE	JEOCIE	EOCAL IE
rw	rw		rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

**Bit 31 INIT:** Initialization mode request

0: Initialization mode is disabled and many control and configuration registers are read only  
 1: Initialization mode has been requested and firmware must wait for INITRDY to become '1' to write to the control and configuration registers  
 When INIT=1, all requests to launch conversions (software, trigger, synchronized) or calibration are ignored.

Bits 30:17 Reserved, must be kept at reset value.

**Bit 17 RDMAEN:** DMA channel enabled to read data for the regular channel

0: The DMA channel is not enabled to read regular data  
 1: The DMA channel is enabled to read regular data  
 RDMAEN must not be '1' if JDMAEN=1.

**Bit 16 JDMAEN:** DMA channel enabled to read data for the injected channel group

0: The DMA channel is not enabled to read injected data  
 1: The DMA channel is enabled to read injected data  
 JDMAEN must not be '1' if RDMAEN=1.

**Bit 15 RSYNC:** Launch regular conversion synchronously with SDADC1

0: Do not launch a regular conversion synchronously with SDADC1  
 1: Launch a regular conversion in this SDADC at the same moment that a regular conversion is launched in SDADC1  
 This bit can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

**Bit 14 JSYNC:** Launch a injected conversion synchronously with SDADC1

0: Do not launch injected conversion synchronously with SDADC1  
 1: Launch an injected conversion in this SDADC at the same moment that an injected conversion is launched in SDADC1  
 This bit can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

Bit 13 Reserved, must be kept at reset value.

**Bit 12 PDI:** Enter power down mode when idle

0: Do not enter power down mode when the SDADC is idle  
 1: Enter Power down when idle  
 When the SDADC is in power down mode due to PDI=1 and a conversion is requested, the SDADC takes 100µs to stabilize before launching the conversion.  
 This bit can be modified only when ADON=0 (SDADC\_CR2).

**Bit 11 SBI:** Enter Standby mode when idle

0: Do not put the SDADC in Standby mode when it is idle

1: Put the SDADC in Standby mode when it is idle

When the SDADC is in Standby mode and a conversion is requested, the SDADC takes 50µs to stabilize before launching the conversion.

Software must not write '1' to SBI at the same time that it writes '1' to PDI.

This bit can be modified only when ADON=0 (SDADC\_CR2).

**Bit 10 SLOWCK:** Slow clock mode enable

0: Disable Slow mode

1: Enable Slow mode (where the SDADC clock frequency should be only 1.5MHz) allowing a lower level of current consumption as well as operation at a lower minimum voltage

This bit may be written only when ADON=0 (SDADC\_CR2).

**Bits 9:8 REFV[1:0]:** Reference voltage selection

00: External reference where the  $V_{REF}$  pin must be forced externally

01: Internal reference where the reference voltage is forced to the 1.2V bandgap voltage internally and the  $V_{REF}$  pin must be connected externally to a capacitance coupled to VSSA

10: Internal reference where the reference voltage is forced to the 1.8V bandgap voltage internally and the  $V_{REF}$  pin must be connected externally to a capacitance coupled to VSSA

11: Internal reference where the reference voltage is forced internally to VDDSDx and the VREFSD+ pin must be externally connected to a capacitance coupled to VREFSD-. See [Section : Constrains on VDDSDx versus VREFSD voltage](#)).

These bits are available only in the register set of SDADC1 and may be written only when ADON=0 (SDADC\_CR2) for all SDADC modules.

**Bit 7:5** Reserved, must be kept at reset value.**Bit 4 ROVRIE:** Regular data overrun interrupt enable

0: Regular data overrun interrupt disabled

1: Regular data overrun interrupt enabled

Please see explanation of ROVRF in SDADC\_ISR.

**Bit 3 REOCIE:** Regular end of conversion interrupt enable

0: Regular end of conversion interrupt disabled

1: Regular end of conversion interrupt enabled

Refer to the description of the REOCF bit in the SDADC\_ISR register.

**Bit 2 JOVRIE:** Injected data overrun interrupt enable

0: Injected data overrun interrupt disabled

1: Injected data overrun interrupt enabled

Refer to the description of the JOVRF bit in the SDADC\_ISR register.

**Bit 1 JEOCIE:** Injected end of conversion interrupt enable

0: Injected end of conversion interrupt disabled

1: Injected end of conversion interrupt enabled

Refer to the description of the JEOCF bit in the SDADC\_ISR register.

**Bit 0 EOCALIE:** End of calibration interrupt enable

0: End of calibration interrupt disabled

1: End of calibration interrupt enabled

Refer to the description of the EOCALF bit in the SDADC\_ISR register.

### 13.6.3 SDADC control register 2 (SDADC\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAST	RSW START	RCONT	Res.	Res.	RCH[3:0]			
							rw	rc_w1	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSW START	JEXTEN[1:0]		Res.	Res.	JEXTSEL[2:0]			Res.	JDS	JCONT	START CALIB	Res.	CALIBCNT[1:0]		ADON
rc_w1	rw	rw			rw	rw	rw		rw	rw	rc_w1		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FAST**: Fast conversion mode selection

- 0: Fast conversion mode disabled
- 1: Fast conversion mode enabled

When converting a single channel in continuous mode, having enabled fast mode causes each conversion (except for the first) to execute 3 times faster (taking 120 SDADC cycles rather than 360). This bit has no effect for conversions which are not continuous.

This bit can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

Bit 23 **RSWSTART**: Software start of a conversion on the regular channel

- 0: Writing '0' has no effect
  - 1: Writing '1' makes a request to start a conversion on the regular channel and causes RCIP to become '1'. If RCIP=1 already or if INIT=1, writing to RSWSTART has no effect
- This bit is always read as '0'.

Bit 22 **RCONT**: Continuous mode selection for regular conversions

- 0: The regular channel is converted just once for each conversion request
  - 1: The regular channel is converted repeatedly after each conversion request
- Writing '0' to this bit while a continuous regular conversion is already in progress stops continuous mode after the conversion already in progress is finished.
- Setting this bit to '1' has no effect on any regular conversion which is pending or already in progress.

Bits 21:20 Reserved, must be kept at reset value.

Bits 19:16 **RCH[3:0]**: Regular channel selection

- 0: Channel 0 is selected as regular channel
- 1: Channel 1 is selected as regular channel
- ...
- 8: Channel 8 is selected as regular channel
- 9-15: Reserved, these values are forbidden

Writing these bits when RCIP=1 takes effect when the next regular conversion begins. This is especially useful in continuous mode (when RCONT=1). It affects also regular conversions which are pending (due to stabilization or due to an ongoing injected conversion).

Bit 15 **JSWSTART**: Start a conversion of the injected group of channels

0: Writing '0' has no effect.

1: Writing '1' makes a request to convert the channels in the injected conversion group, causing JCIP to become '1' at the same time. If JCIP=1 already or if INIT=1, then writing to JSWSTART has no effect.

This bit is always read as '0'.

Bits 14:13 **JEXTEN[1:0]**: Trigger enable and trigger edge selection for injected conversions

00: Trigger detection is disabled

01: Each rising edge on the selected trigger makes a request to launch a injected conversion

10: Each falling edge on the selected trigger makes a request to launch a injected conversion

11: Both rising edges and falling edges on the selected trigger make requests to launch injected conversions

This bit can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

Bit 12:11 Reserved, must be kept at reset value.

Bits 10:8 **JEXTSEL[2:0]**: Trigger signal selection for launching injected conversions

0x0-0x7: Trigger inputs selected by following table.

This bit can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

	SDADC1	SDADC2	SDADC3
0x00	TIM13_CH1	TIM17_CH1	TIM16_CH1
0x01	TIM14_CH1	TIM12_CH1	TIM12_CH2
0x02	TIM15_CH2	TIM2_CH3	TIM2_CH4
0x03	TIM3_CH1	TIM3_CH2	TIM3_CH3
0x04	TIM4_CH1	TIM4_CH2	TIM4_CH3
0x05	TIM19_CH2	TIM19_CH3	TIM19_CH4
0x06	EXTI15	EXTI15	EXTI15
0x07	EXTI11	EXTI11	EXTI11

Bit 7 Reserved, must be kept at reset value.

Bit 6 **JDS**: Delay start of injected conversions.

0: Injected conversions begin as soon as possible after the request

1: After a request for a injected conversion is made, the SDADC waits a fixed interval before launching the conversion, allowing time for any regular conversions which is already in progress to finish, and thus assuring that the timing of the launch is deterministic.

The delay is 500 ADC clocks, unless PDI=1 and SLOWCK=0, in which case the delay is 600 ADC clocks.

This bit can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

Bit 5 **JCONT**: Continuous mode selection for injected conversions

0: The series of conversions which converts each selected channel (the scan sequence) is executed just once for each conversion request

1: The series of conversions for the injected group channels is repeated continuously, starting over with the highest selected channel each time the lowest selected channel finishes its conversion

Writing '0' to this bit while a continuous injected conversion is already in progress stops continuous mode after the conversion already in progress is finished. If an injected conversions is pending or is already in progress when this bit changes to '1', it does not become continuous.

Bit 4 **STARTCALIB**: Start calibration

0: Writing '0' has no effect

1: Writing '1' makes a request to start the calibration sequence, causing CALIBIP to become '1' at the same time

After the request is made, the calibration starts as soon as any ongoing activity (stabilization or a conversion) is finished, or immediately if the SDADC is stabilized and idle.

Writing this bit when CALIBIP=1 or when INIT=1 has no effect.

This bit is always read as '0'.

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **CALIBCNT[1:0]**: Number of calibration sequences to be performed (number of valid configurations)

0: One calibration sequence will be performed to calculate OFFSET0[11:0]

1: Two calibration sequences will be performed to calculate OFFSET0[11:0] and OFFSET1[11:0]

2: Three calibration sequences will be performed to calculate OFFSET0[11:0], OFFSET1[11:0], and OFFSET2[11:0]

3: Reserved, must not use this value

This bit can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

Bit 0 **ADON**: SDADC enable

0: All SDADC functions are disabled. Power down mode is entered, and the flags and the data are cleared

1: SDADC is enabled.

When PDI=0, the SDADC exits power down mode and the 100us of stabilization are observed starting at the moment that ADON is set to '1'.

### 13.6.4 SDADC interrupt and status register (SDADC\_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INITRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STABIP	RCIP	JCIP	CALIBIP	Res.	ROVRF	REOCF	JOVRF	JEOCF	EOCALF						
r	r	r	r								r	r	r	r	r

Bit 31 **INITRDY**: Initialization mode is ready

0: The SDADC is not in initialization mode.

1: The SDADC is in initialization mode

Many control and configuration registers (see their descriptions) can be modified only when INITRDY=1.

Hardware clears this bit as soon as INIT (SDADC\_CR1) is cleared.

Hardware sets this bit after the INIT bit is set. If a conversion or calibration is pending or ongoing when INIT is cleared, INITRDY stays at '0' until all operations have complete. Otherwise, INITRDY becomes '1' about two SDADC clock cycles after INIT is set.

Bits 30:16 Reserved, must be kept at reset value.

Bit 15 **STABIP**: Stabilization in progress status

0: The SDADC is either stabilized or it is in power down mode or Standby mode

1: The SDADC is currently in the process of stabilization, after waking up from either power down mode or Standby mode

A request to start the calibration sequence or to start a conversion can be issued while STABIP=1, with the actions automatically delayed until after stabilization is complete.

Bit 14 **RCIP**: Regular conversion in progress status

0: No request to convert the regular channel has been issued

1: The conversion of the regular channel is in progress or a request for a regular conversion is pending

A request to start a regular conversion is ignored when RCIP=1.

Bit 13 **JCIP**: Injected conversion in progress status

0: No request to convert the injected channel group (neither by software nor by trigger) has been issued

1: The conversion of the injected channel group is in progress or a request for a injected conversion is pending, due either to '1' being written to JSWSTART or to a trigger detection

A request to start a injected conversion is ignored when JCIP=1.

Bit 12 **CALIBIP**: Calibration in progress status

0: No calibration request has been issued

1: Calibration is in progress or a request to start calibration is pending

A request to start calibration is ignored when CALIBIP=1.

Bits 11:5 Reserved, must be kept at reset value.

- Bit 4 **ROVRF**: Regular conversion overrun flag
- 0: No regular conversion overrun has occurred
  - 1: A regular conversion overrun has occurred, which means that a regular conversion finished while REOCF was already '1'. RDATAR is not affected by overruns
- This bit is set by hardware. It can be cleared by software using the CLRROVRF bit in the SDADC\_CLRISR register.
- Bit 3 **REOCF**: End of regular conversion flag
- 0: No regular conversion has completed
  - 1: A regular conversion has completed and its data may be read
- This bit is set by hardware. It is cleared when software reads SDADC\_RDATAR.
- Bit 2 **JOVRF**: Injected conversion overrun flag
- 0: No injected conversion overrun has occurred
  - 1: A injected conversion overrun has occurred, which means that a injected conversion finished while JEOCF was already '1'. JDATAR is not affected by overruns
- This bit is set by hardware. It can be cleared by software using the CLRJOVRF bit in the SDADC\_CLRISR register.
- Bit 1 **JEOCF**: End of injected conversion flag
- 0: No injected conversion has completed
  - 1: A injected conversion has completed and its data may be read
- This bit is set by hardware. It is cleared when software reads SDADC\_JDATAR.
- Bit 0 **EOCALF**: End of calibration flag
- 0: No calibration sequence has completed
  - 1: Calibration has completed and the offsets have been updated
- This bit is set by hardware. It can be cleared by software using the CLREOCALF bit in SDADC\_CLRISR.

*Note:* For each of the flag bits (ROVRF, REOCF, JOVRF, JEOCF, and EOCALF), an interrupt can be enabled by setting the corresponding bit in the SDADC\_CR1 register. If an interrupt is requested, the flag must be cleared before exiting the interrupt service routine.

All the bits of SDADC\_ISR except INITRDY are cleared automatically when ADON=0.

### 13.6.5 SDADC interrupt and status clear register (SDADC\_CLRISR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CLR ROVRF	Res.	CLR JOVRF	Res.	CLR EOCAF										
											rc_w1		rc_w1		rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CLRROVRF**: Clear the regular conversion overrun flag  
 0: Writing '0' has no effect  
 1: Writing '1' clears the ROVRF bit in the SDADC\_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CLRJOVRF**: Clear the injected conversion overrun flag  
 0: Writing '0' has no effect  
 1: Writing '1' clears the JOVRF bit in the SDADC\_ISR register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CLREOCALF**: Clear the end of calibration flag  
 0: Writing '0' has no effect  
 1: Writing '1' clears the EOCAF bit in the SDADC\_ISR register

*Note:* The bits of SDADC\_CLRISR are always read as '0'.

### 13.6.6 SDADC injected channel group selection register (SDADC\_JCHGR)

Address offset: 0x14

Reset value: 0x0000 0001

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

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

Bits 8:0 **JCHG[8:0]**: Injected channel group selection

0: If JCHG[i]=0, then channel i is not part of the injected group (where,  $0 \leq i \leq 8$ )

1: If JCHG[i]=1, then channel i is part of the injected group (where,  $0 \leq i \leq 8$ )

A injected conversion operates always in scan mode, which means that each of selected channels are converted, one after another. The highest channel (channel 8, if selected) is converted first and the sequence ends at the lowest selected channel.

If JCONT=1, this series of conversions is performed continuously.

If JCONT=1, FAST=1, and there is only one channel selected in the injected group, then each of the conversions (besides the first) finishes in only 120 SDADC cycles (rather than 360).

This field can be modified while an injected conversion is in progress and it will take effect for the next group conversion. Writing JCHG also affects injected conversions which are pending (due to stabilization or due to the delay caused by JDS=1).

At least one channel must always be selected for the injected group. Writes causing all JCHG bits to be zero are ignored.

### 13.6.7 SDADC configuration 0 register (SDADC\_CONF0R)

This register specifies the parameters for configuration 0. If CONFCHi[1:0]='00', then each conversion of channel "i" will use the configuration settings specified in this register.

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMMON0[1:0]		Res.	Res.	SE0[1:0]		Res.	Res.	Res.	GAIN0[2:0]			Res.	Res.	Res.	Res.
rw	rw			rw	rw				rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OFFSET0[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **COMMON0[1:0]**: Common mode for configuration 0

- 00: Ground
- 01: VCM (VDD/2)
- 10: VDD
- 11: Reserved, must not use this value

This value is used only during calibration, i.e., when determining the offset. It has no direct effect on the conversions.

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

Bits 27:26 **SE0[1:0]**: Single-ended mode for configuration 0

- 00: Conversions are executed in differential mode
- 01: Conversions are executed in single-ended offset mode
- 10: Reserved, do not use this setting
- 11: Conversions are executed in single-ended zero-volt reference mode

When this field is non-zero, the corresponding negative differential analog input, SDADCx\_AINxM, is connected internally to VSSA so that its pin can be used for other functions.

Bits 25:23 Reserved, must be kept at reset value.

Bits 22:20 **GAIN0[2:0]**: Gain setting for configuration 0

- 000: 1x gain
- 001: 2x gain
- 010: 4x gain
- 011: 8x gain
- 100: 16x gain
- 101: 32x gain
- 111: 0.5x gain

Bits 19:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET0[11:0]**: Twelve-bit calibration offset for configuration 0

For channels which select configuration 0, OFFSET0 is applied to the results of each conversion. This value is automatically set during calibration.

*Note:* This register can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).



### 13.6.8 SDADC configuration 1 register (SDADC\_CONF1R)

This register specifies the parameters for configuration 1. If CONFCHi[1:0]='01', then each conversion of channel "i" will use the configuration settings specified in this register.

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMMON1 [1:0]		Res.	Res.	SE1[1:0]		Res.	Res.	Res.	GAIN1[2:0]			Res.	Res.	Res.	Res.
rw	rw			rw	rw				rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OFFSET1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **COMMON1[1:0]**: Common mode for configuration 1

00: Ground

01: VCM (VDD/2)

10: VDD

11: Reserved, must not use this value

This value is used only during calibration, i.e., when determining the offset. It has no direct effect on the conversions.

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

Bits 27:26 **SE1[1:0]**: Single-ended mode for configuration 1

00: Conversions are executed in differential mode

01: Conversions are executed in single-ended offset mode

10: Reserved, do not use this setting

11: Conversions are executed in single-ended zero-volt reference mode

When this field is non-zero, the corresponding negative differential analog input, INN<sub>x</sub>, is connected internally to VSSA so that its pin can be used for other functions.

Bits 25:23 Reserved, must be kept at reset value.

Bits 22:20 **GAIN1[2:0]**: Gain setting for configuration 1

000: 1x gain

001: 2x gain

010: 4x gain

011: 8x gain

100: 16x gain

101: 32x gain

111: 0.5x gain

Bits 19:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET1[11:0]**: Twelve-bit calibration offset for configuration 1

For channels which select configuration 1, OFFSET1 is applied to the results of each conversion.

This value is automatically set during calibration if CALIBCNT (SDADC\_CR2) has a value greater than or equal to 1.

*Note:* This register can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).

### 13.6.9 SDADC configuration 2 register (SDADC\_CONF2R)

This register specifies the parameters for configuration 2. If CONFCHi[1:0]='10', then each conversion of channel "i" will use the configuration settings specified in this register.

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMMON2 [1:0]		Res.	Res.	SE2[1:0]		Res.	Res.	Res.	GAIN2[2:0]			Res.	Res.	Res.	Res.
rw	rw			rw	rw				rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OFFSET2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **COMMON2[1:0]**: Common mode for configuration 2  
 00: VSSSD  
 01: VDDSDx/2  
 10: VDDSDx  
 11: Reserved, this value is forbidden  
 This value is used only during calibration, i.e., when determining the offset. It has no direct effect on the conversions.

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

Bits 27:26 **SE2[1:0]**: Single-ended mode for configuration 2  
 00: Conversions are executed in differential mode  
 01: Conversions are executed in single-ended offset mode  
 10: Reserved, do not use this setting  
 11: Conversions are executed in single-ended zero-volt reference mode  
 When this field is non-zero, the corresponding negative differential analog input, INN<sub>x</sub>, is connected internally to VSSA so that its pin can be used for other functions.

Bits 25:23 Reserved, must be kept at reset value.

Bits 22:20 **GAIN2[2:0]**: Gain setting for configuration 2  
 000: 1x gain  
 001: 2x gain  
 010: 4x gain  
 011: 8x gain  
 100: 16x gain  
 101: 32x gain  
 111: 0.5x gain

Bits 19:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET2[11:0]**: Twelve-bit calibration offset for configuration 2  
 For channels which select configuration 2, OFFSET2 is applied to the results of each conversion. This value is automatically set during calibration if CALIBCNT (SDADC\_CR2) has a value greater than or equal to 2.

*Note:* This register can be modified only when INITRDY=1 (SDADC\_ISR) or ADON=0 (SDADC\_CR2).



### 13.6.10 SDADC channel configuration register 1 (SDADC\_CONFCHR1)

This register specifies which configurations are to be used by channels 0-7.

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CONFCH7[1:0]		Res.	Res.	CONFCH6[1:0]		Res.	Res.	CONFCH5[1:0]		Res.	Res.	CONFCH4[1:0]	
		rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CONFCH3[1:0]		Res.	Res.	CONFCH2[1:0]		Res.	Res.	CONFCH1[1:0]		Res.	Res.	CONFCH0[1:0]	
		rw	rw												

**CONFCHi[1:0]:** Channel i configuration

- 00: Channel i uses the configuration specified in SDADC\_CONF0R
- 01: Channel i uses the configuration specified in SDADC\_CONF1R
- 10: Channel i uses the configuration specified in SDADC\_CONF2R
- 11: Reserved, must not use this value

*Note:* This register can be modified only when *INITRDY=1 (SDADC\_ISR)* or *ADON=0 (SDADC\_CR2)*.

### 13.6.11 SDADC channel configuration register 2 (SDADC\_CONFCHR2)

This register specifies which configuration is to be used by channel 8.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															CONFCH8[1:0]
														rw	rw

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

Bits 1:0 **CONFCH8[1:0]:** Channel 8 configuration

- 00: Channel 8 uses the configuration specified in SDADC\_CONF0R
- 01: Channel 8 uses the configuration specified in SDADC\_CONF1R
- 10: Channel 8 uses the configuration specified in SDADC\_CONF2R
- 11: Reserved, must not use this value

*Note:* This register can be modified only when *INITRDY=1 (SDADC\_ISR)* or *ADON=0 (SDADC\_CR2)*.

### 13.6.12 SDADC data register for injected group (SDADC\_JDATAR)

This register contains the data resulting from the recently completed conversion of a channel in the injected group.

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	JDATACH[3:0]				Res.							
				r	r	r	r								
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 31:28 Reserved, must be kept at reset value.

Bits 27:24 **JDATACH[3:0]**: Injected channel most recently converted

When each conversion of a channel in the injected group finishes, JDATACH[3:0] is updated to indicate which channel was converted. This field is valid when JEOCF=1, and is set to zero when JEOCF is cleared. Thus, when JEOCF=1, JDATA[15:0] holds the data that corresponds to the channel indicated by JDATACH[3:0].

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **JDATA[15:0]**: Injected group conversion data

When each conversion of a channel in the injected group finishes, its resulting data is stored in this field. The data is valid when JEOCF=1. Reading this register clears both this field as well as the corresponding JEOCF.

*Note:* DMA may be used to read the data from this register. Half-word accesses may be used to read only the conversion data.

*Note:* This register is cleared as soon as it is read. Reading this register also clears JEOCF in SDADC\_ISR. Thus, firmware must not read this register if DMA is activated to read data from this register.

### 13.6.13 SDADC data register for the regular channel (SDADC\_RDATAR)

This register contains the data resulting from the recently completed conversion of the regular channel.

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[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 **RDATA[15:0]**: Regular channel conversion data

When each regular conversion finishes, its data is stored in this register. The data is valid when REOCF=1. Reading this register clears both this field as well as the corresponding JEOCF.

*Note:* This register is cleared as soon as it is read. Reading this register also clears REOCF in SDADC\_ISR.

**13.6.14 SDADC1 and SDADC2 injected data register (SDADC\_JDATA12R)**

This register contains the data resulting from the recently completed conversions of injected channels of SDADC1 and SDADC2. The data is a mirror image of the data in the corresponding SDADC\_JDATAR registers. This register is available only in the set of registers for SDADC1 and must not be accessed unless the JSYNC bit of SDADC2 is set.

Address offset: 0x70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA2[15:0]															
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
JDATA1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

**Bits 31:16 JDATA2[15:0]:** Injected group conversion data for SDADC2  
 When each conversion of a channel in the injected group of SDADC2 finishes, its resulting data is stored in this field. The data is valid only when JEOCF of SDADC2 is set. Reading this register clears both this field as well as the corresponding JEOCF.

**Bits 15:0 JDATA1[15:0]:** Injected group conversion data for SDADC1  
 When each conversion of a channel in the injected group of SDADC1 finishes, its resulting data is stored in this field. The data is valid only when JEOCF of SDADC1 is set. Reading this register clears both this field as well as the corresponding JEOCF.

*Note:* DMA may be used to read the data from this register, in which case 32-bit word accesses must be used.

*Note:* This register is cleared as soon as it is read. Reading this register also clears JEOCF in the corresponding two SDADC\_ISR registers. Thus, firmware must not read this register nor the SDADC\_JDATA registers of SDADC1 and SDADC2 if DMA is activated to read data from this register.

### 13.6.15 SDADC1 and SDADC2 regular data register (SDADC\_RDATA12R)

This register contains the data resulting from the recently completed conversions of regular channels of SDADC1 and SDADC2. The data is a mirror image of the data in the corresponding SDADC\_RDATAR registers. This register is available only in the set of registers for SDADC1 and must not be accessed unless the RSYNC bit of SDADC2 is set.

Address offset: 0x74

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA2[15:0]															
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
RDATA1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA2[15:0]**: Regular conversion data for SDADC2

When each conversion of the regular channel of SDADC2 finishes, its resulting data is stored in this field. The data is valid only when REOCF of SDADC2 is set. Reading this register clears both this field as well as the corresponding REOCF.

Bits 15:0 **RDATA1[15:0]**: Regular conversion data for SDADC1

When each conversion of the regular channel of SDADC1 finishes, its resulting data is stored in this field. The data is valid only when REOCF of SDADC1 is set. Reading this register clears both this field as well as the corresponding REOCF.

*Note:* DMA may be used to read the data from this register, in which case 32-bit word accesses must be used.

*Note:* This register is cleared as soon as it is read. Reading this register also clears REOCF in the corresponding two SDADC\_ISR registers. Thus, firmware must not read this register nor the SDADC\_RDATA registers of SDADC1 and SDADC2 if DMA is activated to read data from this register.

**13.6.16 SDADC1 and SDADC3 injected data register (SDADC\_JDATA13R)**

This register contains the data resulting from the recently completed conversions of injected channels of SDADC1 and SDADC3. The data is a mirror image of the data in the corresponding SDADC\_JDATAR registers. This register is available only in the set of registers for SDADC1 and must not be accessed unless the JSYNC bit of SDADC3 is set.

Address offset: 0x78

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA3[15:0]															
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
JDATA1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **JDATA3[15:0]**: Injected group conversion data for SDADC3  
 When each conversion of a channel in the injected group of SDADC3 finishes, its resulting data is stored in this field. The data is valid only when JEOCF of SDADC3 is set. Reading this register clears both this field as well as the corresponding JEOCF.

Bits 15:0 **JDATA1[15:0]**: Injected group conversion data for SDADC1  
 When each conversion of a channel in the injected group of SDADC1 finishes, its resulting data is stored in this field. The data is valid only when JEOCF of SDADC1 is set. Reading this register clears both this field as well as the corresponding JEOCF.

*Note:* DMA may be used to read the data from this register, in which case 32-bit word accesses must be used.

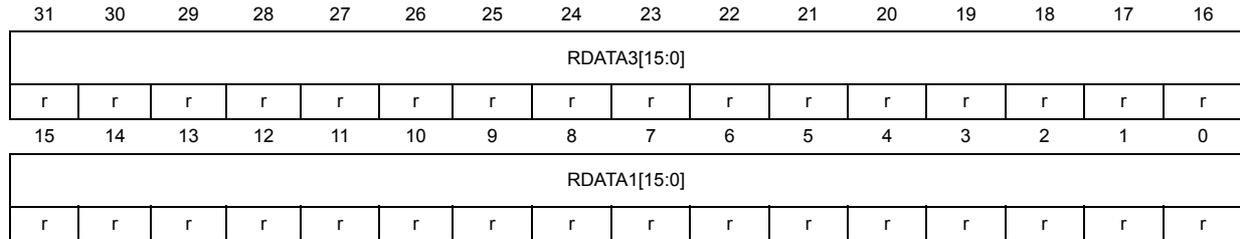
*Note:* This register is cleared as soon as it is read. Reading this register also clears JEOCF in the corresponding two SDADC\_ISR registers. Thus, firmware must not read this register nor the SDADC\_JDATA registers of SDADC1 and SDADC3 if DMA is activated to read data from this register.

### 13.6.17 SDADC1 and SDADC3 regular data register (SDADC\_RDATA13R)

This register contains the data resulting from the recently completed conversions of regular channels of SDADC1 and SDADC3. The data is a mirror image of the data in the corresponding SDADC\_RDATAR registers. This register is available only in the set of registers for SDADC1 and must not be accessed unless the RSYNC bit of SDADC3 is set.

Address offset: 0x7C

Reset value: 0x0000 0000



Bits 31:16 **RDATA3[15:0]**: Regular conversion data for SDADC3

When each conversion of the regular channel of SDADC3 finishes, its resulting data is stored in this field. The data is valid only when REOCF of SDADC3 is set. Reading this register clears both this field as well as the corresponding REOCF.

Bits 15:0 **RDATA1[15:0]**: Regular conversion data for SDADC1

When each conversion of the regular channel of SDADC1 finishes, its resulting data is stored in this field. The data is valid only when REOCF of SDADC1 is set. Reading this register clears both this field as well as the corresponding REOCF.

*Note:* DMA may be used to read the data from this register, in which case 32-bit word accesses must be used.

*Note:* This register is cleared as soon as it is read. Reading this register also clears REOCF in the corresponding two SDADC\_ISR registers. Thus, firmware must not read this register nor the SDADC\_RDATA registers of SDADC1 and SDADC3 if DMA is activated to read data from this register.

13.6.18 SDADC register map

The following table summarizes the ADC registers.

Table 37. SDADC 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	SDADC_CR1	INIT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDMAEN	JDMAEN	RSYNC (1)	JSYNC (1)	Res.	Res.	SBI	SLOWCK	REFV[1:0] <sup>(2)</sup>	Res.	Res.	Res.	Res.	ROVRIE	REOCIE	JOVRIE	JEOCIE	EOCALIE	
	Reset value	0														0	0	0	0			0	0	0	0				0	0	0	0	0	
0x04	SDADC_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FAST	RSWSTART	RCONT	Res.	Res.	RCH [3:0]			JSWSTART	JEXTEN	Res.	JEXTSEL [3:0]			Res.	JDS	JCONT	STARTCALIB	Res.	CALIB CNT [1:0]	ADON					
	Reset value								0	0	0			0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	
0x08	SDADC_ISR	INITRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STABIP	RCIP	JCIP	CALBIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROVRF	REOCF	JOVRF	JEOCF	EOCALF	
	Reset value	0																0	0	0	0							0	0	0	0	0	0	
0x0C	SDADC_CLRISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRROVRF	Res.	CLRJOVRF	Res.	CLREOCALF		
	Reset value																											0	0	0	0	0	0	
0x10	Res.																																	
0x14	SDADC_JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[8:0]	
	Reset value																																1	
0x18	Res.																																	
0x1C	Res.																																	
0x20	SDADC_CONF0R	COM MON 0 [1:0]	Res.	Res.	SE0 [1:0]	Res.	Res.	Res.	GAIN0 [2:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET0[11:0]
	Reset value	0	0		0	0			0	0	0																		0	0	0	0	0	0
0x24	SDADC_CONF1R	COM MON 1 [1:0]	Res.	Res.	SE1 [1:0]	Res.	Res.	Res.	GAIN1 [2:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[11:0]
	Reset value	0	0		0	0			0	0	0																		0	0	0	0	0	0
0x28	SDADC_CONF2R	COM MON 2 [1:0]	Res.	Res.	SE2 [1:0]	Res.	Res.	Res.	GAIN2 [2:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[11:0]
	Reset value	0	0		0	0			0	0	0																		0	0	0	0	0	0
0x2C - 0x3C	Res.																																	



## 14 Digital-to-analog converter (DAC1 and DAC2)

### 14.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. An input reference voltage,  $V_{DDA}$  (shared with ADC), is available. The output can optionally be buffered for higher current drive.

### 14.2 DAC1/2 main features

The devices integrate three 12-bit DAC channels:

- DAC1 integrates two DAC channels:
  - DAC1 channel 1 which output is DAC1\_OUT1
  - DAC1 channel 2 which output is DAC1\_OUT2The two channels can be used independently or simultaneously when both channels are grouped together for synchronous update operations (dual mode).
- DAC2 integrates only one channel, DAC2 channel 1 which output is DAC2\_OUT1 .

The DAC main features are the following:

- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation (DAC1 only)
- Triangular-wave generation (DAC1 only)
- Independent or simultaneous conversions (dual mode only)
- DMA capability for each channel
- DMA underrun error detection
- External triggers for conversion
- Programmable internal buffer
- Input voltage reference,  $V_{DDA}$

[Figure 41](#) and [Figure 42](#) show the block diagram of a DAC1 and DAC2 channels and [Table 38](#) gives the pin description.

Figure 41. DAC1 block diagram

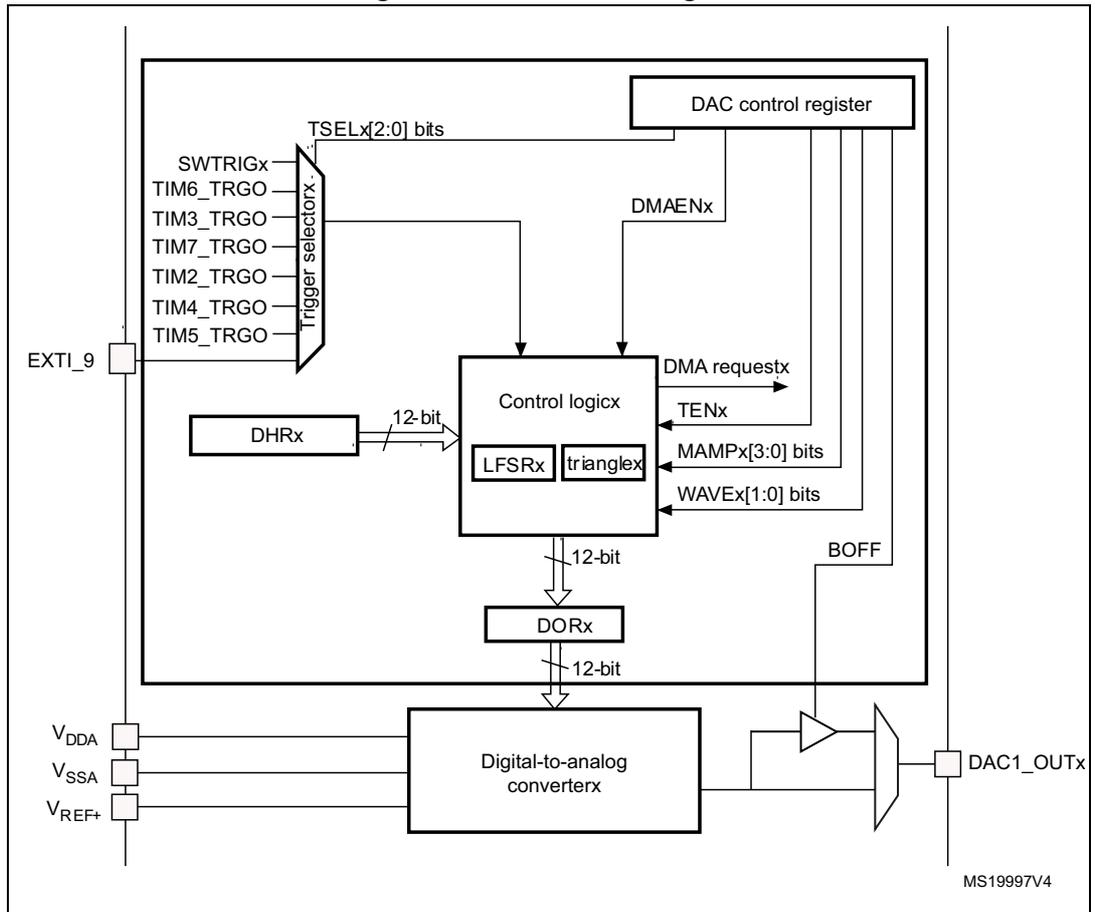


Figure 42. DAC2 block diagram

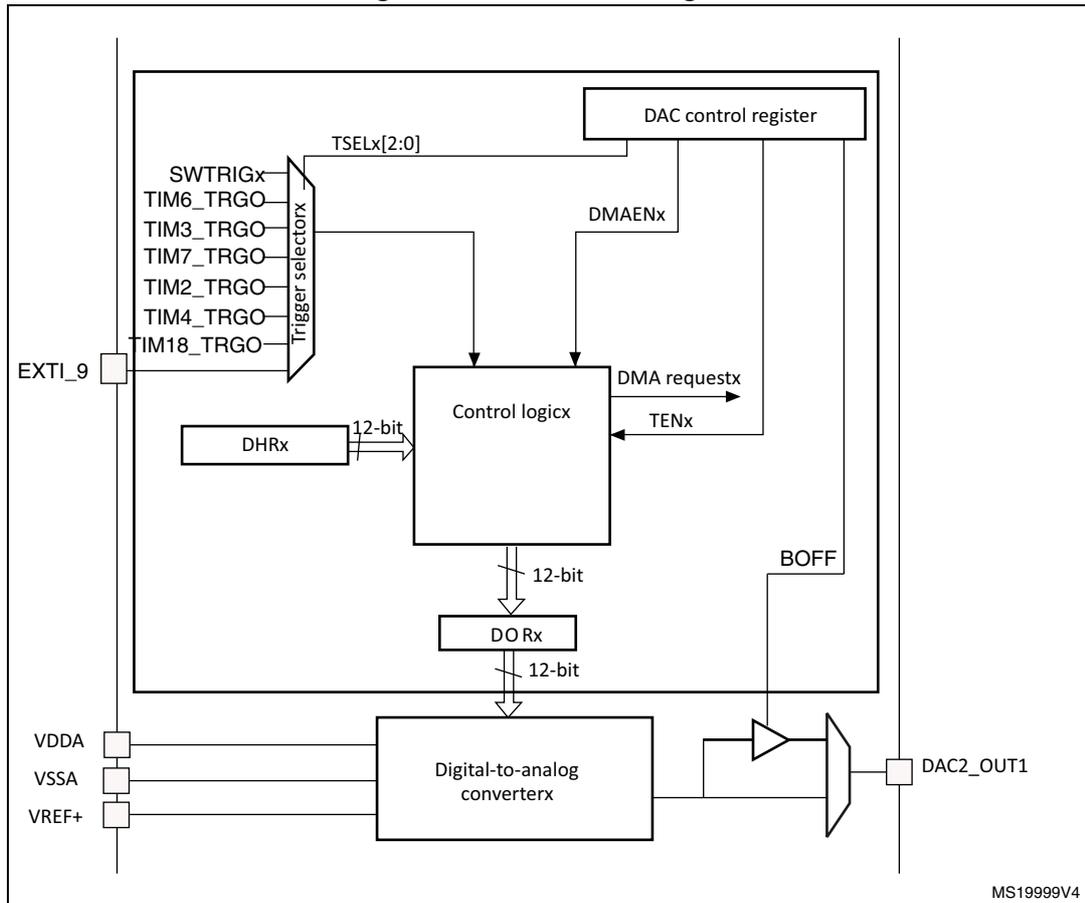


Table 38. DAC1/2 pins

Name	Signal type	Remarks
V <sub>REF+</sub>	Input, analog reference positive	The higher/positive reference voltage for the DAC, $1.8\text{ V} \leq V_{\text{REF}+} \leq V_{\text{DDA}}$
V <sub>DDA</sub>	Input, analog supply	Analog power supply
V <sub>SSA</sub>	Input, analog supply ground	Ground for analog power supply
DAC1_OUT1	Analog output signal	DACx channel y analog output

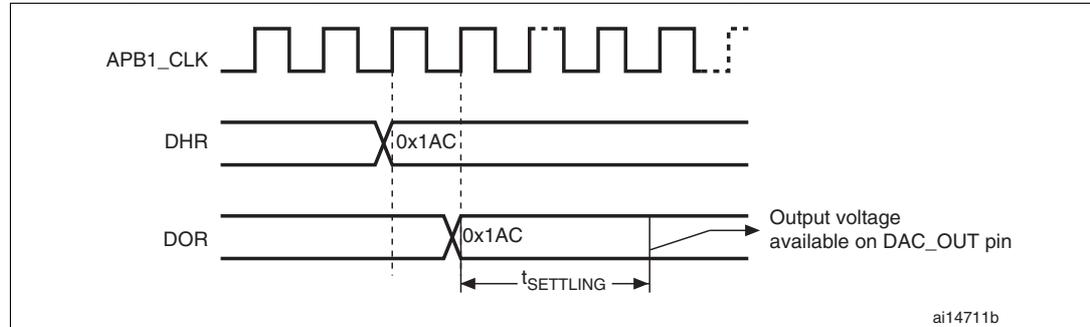
*Note:* Once the DACx channel y is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DACx\_OUTy). In order to avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN).



register is set) and a trigger occurs, the transfer is performed three PCLK1 clock cycles later.

When DAC\_DORx is loaded with the DAC\_DHRx contents, the analog output voltage becomes available after a time  $t_{\text{SETTLING}}$  that depends on the power supply voltage and the analog output load.

**Figure 44. Timing diagram for conversion with trigger disabled  $TEN = 0$**



### Independent trigger with single LFSR generation

To configure the DAC in this conversion mode (see [Section 14.5: Noise generation](#)), the following sequence is required:

1. Set the DAC channel trigger enable bit  $TENx$ .
2. Configure the trigger source by setting  $TSELx[2:0]$  bits.
3. Configure the DAC channel  $WAVEx[1:0]$  bits as "01" and the same LFSR mask value in the  $MAMPx[3:0]$  bits
4. Load the DAC channel data into the desired DAC\_DHRx register (DHR12RD, DHR12LD or DHR8RD).

When a DAC channelx trigger arrives, the LFSRx counter, with the same mask, is added to the DHRx register and the sum is transferred into DAC\_DORx (three APB clock cycles later). Then the LFSRx counter is updated.

### Independent trigger with single triangle generation

To configure the DAC in this conversion mode (see [Section 14.6: Triangle-wave generation](#)), the following sequence is required:

1. Set the DAC channelx trigger enable  $TENx$  bits.
2. Configure the trigger source by setting  $TSELx[2:0]$  bits.
3. Configure the DAC channelx  $WAVEx[1:0]$  bits as "1x" and the same maximum amplitude value in the  $MAMPx[3:0]$  bits
4. Load the DAC channelx data into the desired DAC\_DHRx register. (DHR12RD, DHR12LD or DHR8RD).

When a DAC channelx trigger arrives, the DAC channelx triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC\_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

### 14.3.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and VDDA.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{\text{DDA}} \times \frac{\text{DOR}}{4095}$$

### 14.3.6 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 40](#).

**Table 39. External triggers (DAC1)**

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 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

**Table 40. External triggers (DAC2)**

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 18 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:* TSELx[2:0] bit cannot be changed when the ENx bit is set. When software trigger is selected, the transfer from the DAC\_DHRx register to the DAC\_DORx register takes only one APB1 clock cycle.

## 14.4 Dual mode functional description

### 14.4.1 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.

### 14.4.2 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.

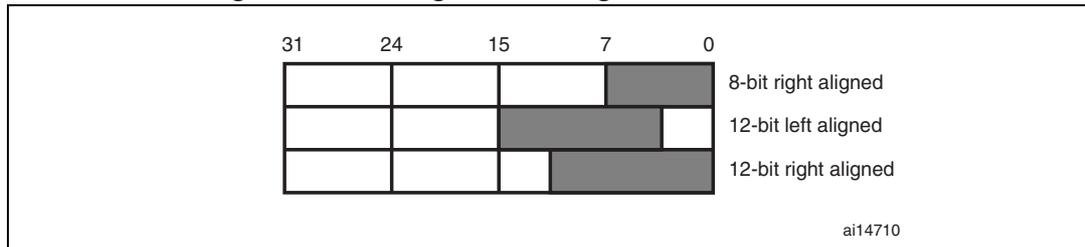
### 14.4.3 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel mode
  - 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\_DHRyyx 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.

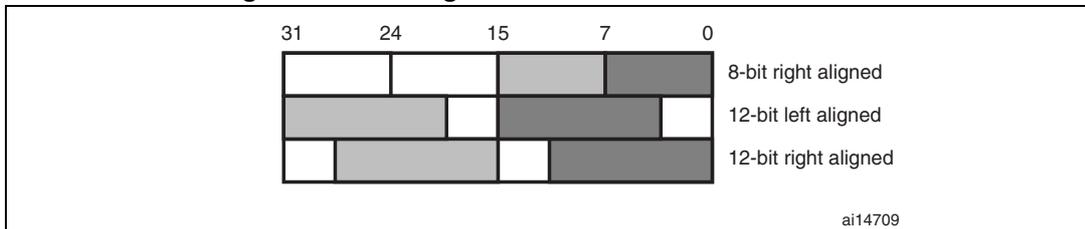
Figure 45. Data registers in single DAC channel mode



- Dual DAC channel mode, there are three possibilities:
  - 8-bit right alignment: data for DAC channel1 to be loaded in the DAC\_DHR8RD [7:0] bits (stored in the DHR1[11:4] bits) and data for DAC channel2 to be loaded in the DAC\_DHR8RD [15:8] bits (stored in 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 in the DHR2[11:0] bits)
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC\_DHR12RD [11:0] bits (stored in the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC\_DHR12LD [27:16] bits (stored in the DHR2[11:0] bits)

Depending on the loaded DAC\_DHRyyyD register, the data written by the user is shifted and stored in 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 46. Data registers in dual DAC channel mode



#### 14.4.4 DAC channel conversion in dual mode

The DAC channel conversion in dual mode is performed in the same way as in single mode (refer to [Section 14.3.4](#)) except that the data have to be loaded by writing to DAC\_DHR8Rx, DAC\_DHR12Lx, DAC\_DHR12Rx, DAC\_DHR8RD, DAC\_DHR12LD or DAC\_DHR12RD.

#### 14.4.5 Description of dual conversion modes

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.

Refer to [Section 14.3.4: DAC channel conversion](#) for details on the APB bus (APB or APB1) that clocks the DAC conversions.

### Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
3. 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 APB clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC\_DOR2 (three APB clock cycles later).

### Independent trigger with single LFSR generation

To configure the DAC in this conversion mode (refer to [Section 14.5: Noise generation](#)), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as "01" and the same LFSR mask value in the MAMPx[3:0] bits
4. Load the dual DAC channel data into the desired DHR register (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 APB 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 APB clock cycles later). Then the LFSR2 counter is updated.

### Independent trigger with different LFSR generation

To configure the DAC in this conversion mode (refer to [Section 14.5: Noise generation](#)), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as "01" and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

When a DAC 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 APB 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 APB clock cycles later). Then the LFSR2 counter is updated.

### Independent trigger with single triangle generation

To configure the DAC in this conversion mode (refer to [Section 14.6: Triangle-wave generation](#)), the following sequence is required:

1. Set the DAC channelx trigger enable TENx bits.
2. Configure different trigger sources by setting different values in the TSELx[2:0] bits
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and the same maximum amplitude value in the MAMPx[3:0] bits
4. Load the DAC channelx data into the desired DAC\_DHRx register.

Refer to [Section 14.3.4: DAC channel conversion](#) for details on the APB bus (APB or APB1) that clocks the DAC conversions.

When a DAC channelx trigger arrives, the DAC channelx triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC\_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

### Independent trigger with different triangle generation

To configure the DAC in this conversion mode (refer to [Section 14.6: Triangle-wave generation](#)), the following sequence is required:

1. Set the DAC channelx trigger enable TENx bits.
2. Configure different trigger sources by setting different values in the TSELx[2:0] bits
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMPx[3:0] bits
4. Load the DAC channelx data into the desired DAC\_DHRx register.

When a DAC channelx trigger arrives, the DAC channelx triangle counter, with a triangle amplitude configured by MAMPx[3:0], is added to the DHRx register and the sum is transferred into DAC\_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

### Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

1. Load the dual DAC channel data to the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

In this configuration, one APB clock cycles).

### Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
3. Load the dual DAC channel data to the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC\_DOR1 and DAC\_DOR2, respectively (after three APB clock cycles).

### Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode (refer to [Section 14.5: Noise generation](#)), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
4. Load the dual DAC channel data 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 APB 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 APB clock cycles later). The LFSR2 counter is then updated.

### Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode (refer to [Section 14.5: Noise generation](#)), the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
3. Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB 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 APB clock cycles later). The LFSR2 counter is then updated.

### Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode (refer to [Section 14.6: Triangle-wave generation](#)), the following sequence is required:

1. Set the DAC channelx trigger enable TEN1x bits.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSELx[2:0] bits.
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and the same maximum amplitude value using the MAMPx[3:0] bits
4. Load the DAC channelx data into the desired DAC\_DHRx registers.

When a trigger arrives, the DAC channelx triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC\_DORx (three APB clock cycles later). The DAC channelx triangle counter is then updated.

**Simultaneous trigger with different triangle generation**

To configure the DAC in this conversion mode ‘refer to [Section 14.6: Triangle-wave generation](#)), the following sequence is required:

1. Set the DAC channelx trigger enable TENx bits.
2. Configure the same trigger source for DAC channelx by setting the same value in the TSELx[2:0] bits
3. Configure the DAC channelx WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMPx[3:0] bits.
4. Load the DAC channelx data into the desired DAC\_DHRx registers.

When a trigger arrives, the DAC channelx triangle counter, with a triangle amplitude configured by MAMPx[3:0], is added to the DHRx register and the sum is transferred into DAC\_DORx (three APB clock cycles later). Then the DAC channelx triangle counter is updated.

**14.4.6 DAC output voltage**

Digital inputs are converted to output voltages on a linear conversion between 0 and VDDA.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$DAC_{output} = V_{DDA} \times \frac{DOR}{4095}$$

**14.4.7 DAC trigger selection**

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which out of 8 possible events will trigger conversion as shown in table below.

**Table 41. External triggers (DAC1)**

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 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 APB cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC\_DORx register has been loaded with the DAC\_DHRx register contents.

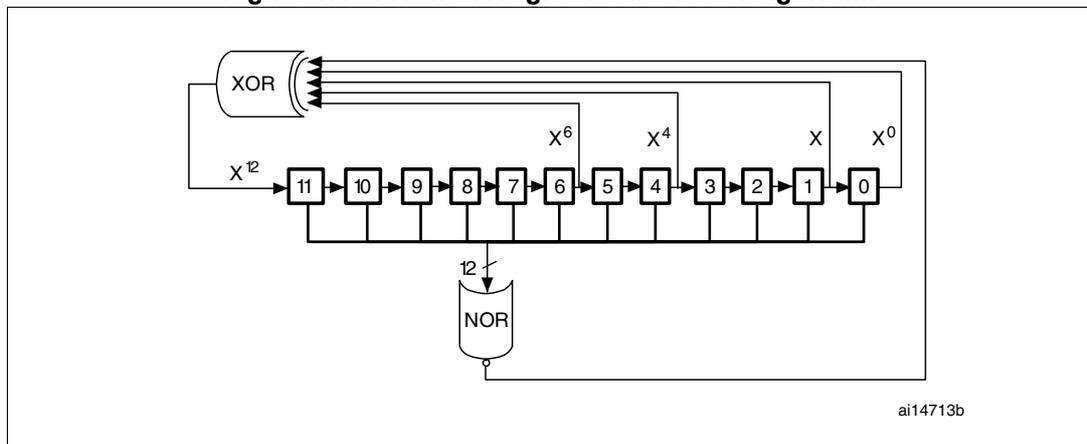
*Note: TSELx[2:0] bit cannot be changed when the ENx bit is set.*

*When software trigger is selected, the transfer from the DAC\_DHRx register to the DAC\_DORx register takes only one APB clock cycle.*

### 14.5 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 APB clock cycles after each trigger event, following a specific calculation algorithm.

**Figure 47. DAC LFSR register calculation algorithm**

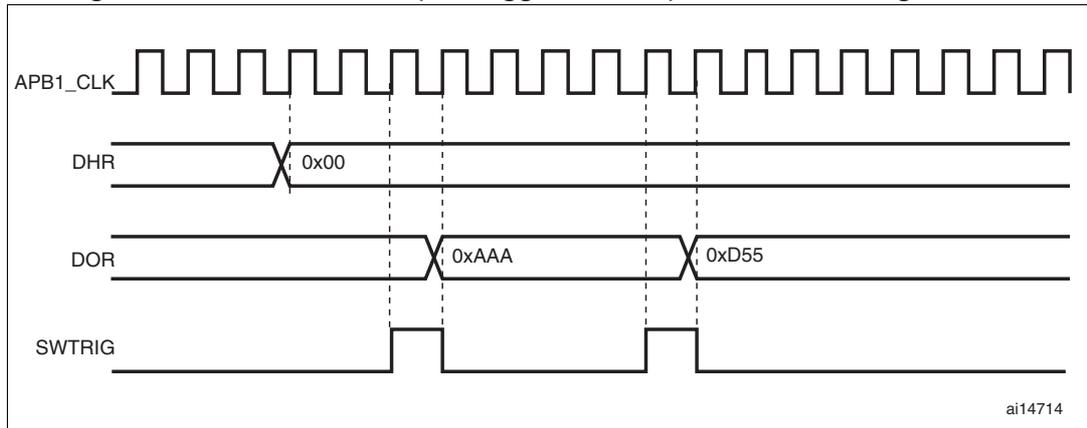


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC\_CR register, is added up to the DAC\_DHRx contents without overflow and this value is then stored into the DAC\_DORx register.

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 48. 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.

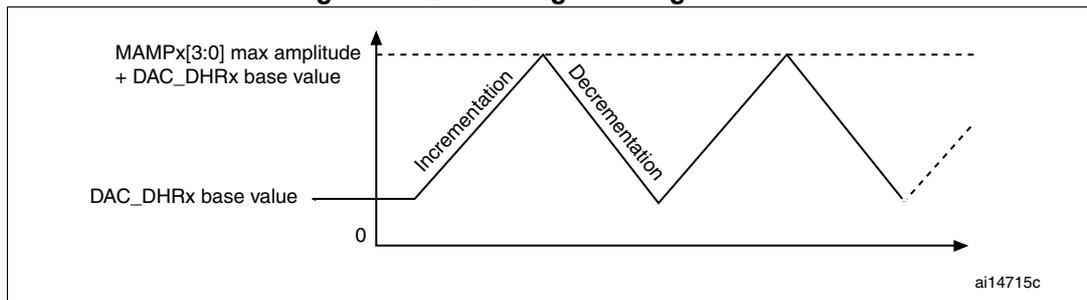
Noise generation is not available on DAC2.

## 14.6 Triangle-wave generation

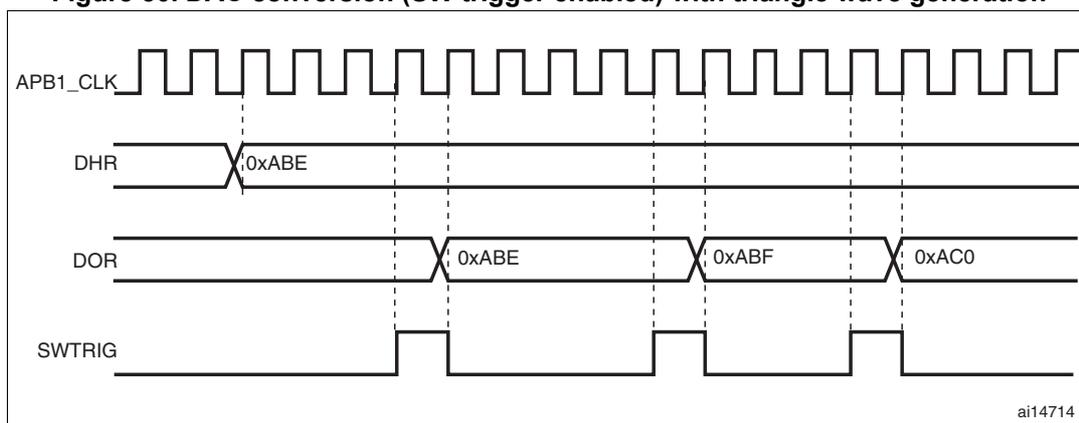
It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting *WAVEx[1:0]* to “10”. The amplitude is configured through the *MAMPx[3:0]* bits in the *DAC\_CR* register. An internal triangle counter is incremented three APB 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 49. DAC triangle wave generation**



**Figure 50. DAC conversion (SW trigger enabled) with triangle wave generation**



*Note:* The DAC trigger must be enabled for triangle generation by setting the *TENx* bit in the *DAC\_CR* register.  
 The *MAMPx[3:0]* bits must be configured before enabling the DAC, otherwise they cannot be changed.  
 Triangle-wave generation is not available on DAC2.

## 14.7 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 to 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 acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag *DMAUDRx* in the *DAC\_SR* register is set, reporting the error condition. 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. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channel, an interrupt is also generated if the corresponding *DMAUDRIEx* bit in the *DAC\_CR* register is enabled.

## 14.8 DAC registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

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

### 14.8.1 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
Res.	Res.	DMAU DRIE2	DMA EN2	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
Res.	Res.	DMAU DRIE1	DMA EN1	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, must be kept at reset value.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

*Note: This bit is available in dual mode only. It is reserved in single mode.*

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

*Note: This bit is available in dual mode only. It is reserved in single mode.*

Bits 27:24 **MAMP2[3:0]**: DAC1 channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

*Note: These bits are available in dual mode only. They are reserved in single mode.*

- Bits 23:22 **WAVE2[1:0]**: DAC1 channel2 noise/triangle wave generation enable  
These bits are set/reset by software.  
00: wave generation disabled  
01: Noise wave generation enabled  
1x: Triangle wave generation enabled  
*Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)*  
*These bits are available in dual mode only. They are reserved in single mode.*
- 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 TRGO event  
100: Timer 2 TRGO event  
101: Timer 4 TRGO event  
110: EXTI line9  
111: Software trigger  
*Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)*  
*These bits are available in dual mode only. They are reserved in single mode.*
- 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 register to the DAC\_DOR2 register takes only one APB1 clock cycle.*  
*Note: This bit is available in dual mode only. It is reserved in single mode.*
- 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  
*Note: This bit is available in dual mode only. It is reserved in single mode.*
- Bit 16 **EN2**: DAC channel2 enable  
This bit is set and cleared by software to enable/disable DAC channel2.  
0: DAC channel2 disabled  
1: DAC channel2 enabled  
*Note: This bit is available in dual mode only. It is reserved in single mode.*
- Bits 15:14 Reserved, must be kept at reset value.
- 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]**: DAC1 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]**: DAC1 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 TRGO event (for DAC1), Timer 18 TRGO event (for DAC2)

100: Timer 2 TRGO event

101: Timer 4 TRGO event

110: EXTI 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

### 14.8.2 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
Res.	Res.														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SWTRIG2	SWTRIG1													
														w	w

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

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.*

*This bit is available in dual mode only. It is reserved in single mode.*

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.*

### 14.8.3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

### 14.8.4 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACCDHR[11:0]												Res.	Res.	Res.	Res.
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w				

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data  
 These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

### 14.8.5 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DACCDHR[7:0]														
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data  
 These bits are written by software which specifies 8-bit data for DAC channel1.

### 14.8.6 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACCDHR[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w



Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

### 14.8.7 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

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, must be kept at reset value.

### 14.8.8 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DACC2DHR[7:0]														
								rw	rw	rw	rw	rw	rw	rw	rw

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

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.

### 14.8.9 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
Res.	Res.	Res.	Res.	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
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data  
 These bits are written by software which specifies 12-bit data for DAC channel1.

### 14.8.10 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]												Res.	Res.	Res.	Res.
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]												Res.	Res.	Res.	Res.
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, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data  
 These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

### 14.8.11 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:168 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel1.

### 14.8.12 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.

### 14.8.13 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

### 14.8.14 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
Res.	Res.	DMAUDR2	Res.												
		rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMAUDR1	Res.												
		rc_w1													

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

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate)

*Note: This bit is available in dual mode only. It is reserved in single mode.*

Bits 28:14 Reserved, must be kept at reset value.

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bits 12:0 Reserved, must be kept at reset value.

14.8.15 DAC register map

Table 42 summarizes the DAC registers.

Table 42. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	Res.	DMAUDRIE2	DMAEN2	MAMP2[3:0]			WAVE2[1:0]			TSEL2[2:0]		TEN2	BOFF2	EN2	Res.	Res.	DMAUDRIE1	DMAEN1	MAMP1[3:0]			WAVE1[1:0]			TSEL1[2:0]		TEN1	BOFF1	EN1		
	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
0x04	DAC_SWTRIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																														0	0	
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC1DHR[11:0]							Res.	Res.	Res.	Res.		
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_DHR12L1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC1DHR[11:0]							Res.	Res.	Res.	Res.		
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC1DHR[7:0]				Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x18	DAC_DHR12L2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x20	DAC_DHR12RD	Res.	Res.	Res.	Res.	DAC2DHR[11:0]							Res.	Res.	Res.	Res.	DAC1DHR[11:0]							Res.	Res.	Res.	Res.						
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_DHR12LD	DAC2DHR[11:0]							Res.	Res.	Res.	Res.	DAC1DHR[11:0]							Res.	Res.	Res.	Res.										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x28	DAC_DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC2DHR[7:0]				DAC1DHR[7:0]				Res.	Res.	Res.	Res.	
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC1DOR[11:0]							Res.	Res.	Res.	Res.	
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAC2DOR[11:0]							Res.	Res.	Res.	Res.	
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	



Table 42. DAC register map (continued) and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	DAC_SR	Res.	Res.	DMAUDR2	Res.	DMAUDR1	Res.																										
	Reset value			0																0													

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 15 Comparator (COMP)

### 15.1 Introduction

STM32F37xx devices embed two general purpose comparators COMP1 and COMP2, that can be used either as standalone devices (all terminal are available on I/Os) or combined with the timers. They can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with the DAC and a PWM output from a timer.

### 15.2 COMP main features

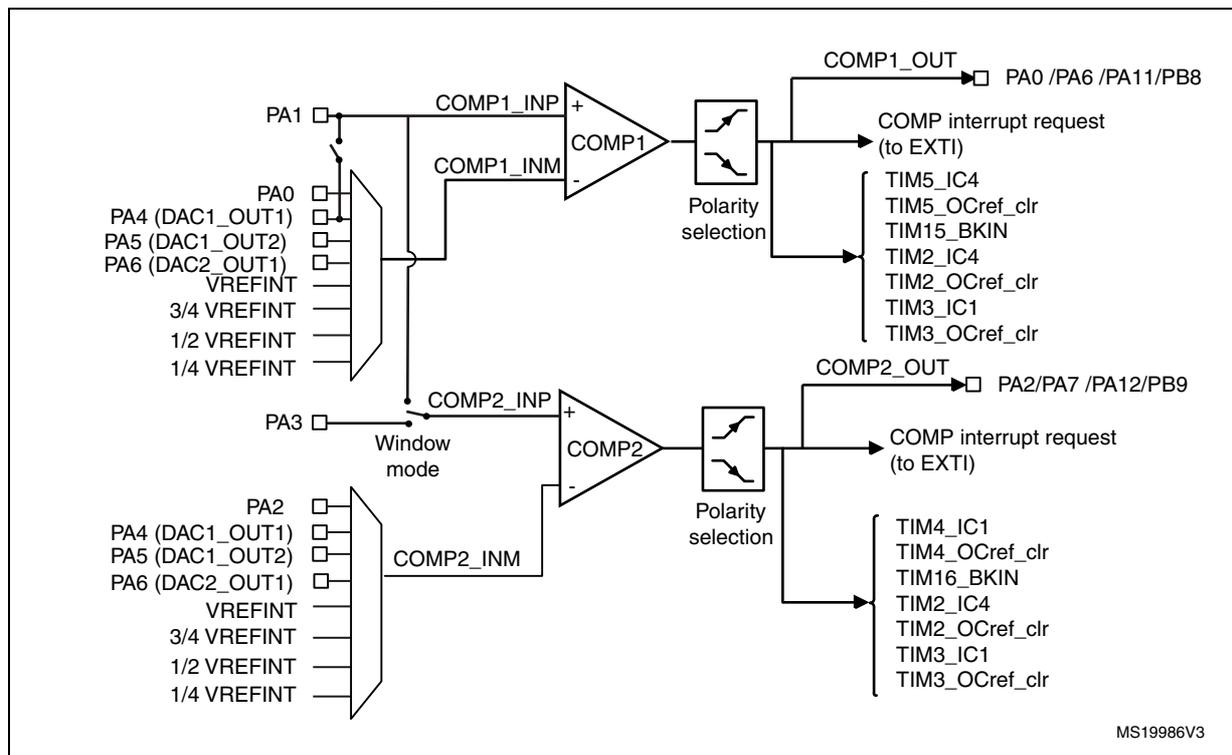
- Rail-to-rail comparators
- Each comparator has positive and configurable negative inputs used for flexible voltage selection:
  - 3 I/O pins
  - DAC1
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to multiple timer inputs for triggering:
  - Capture events
  - OCREF\_CLR events (for cycle-by-cycle current control)
  - Break events for fast PWM shutdowns
- COMP1 and COMP2 comparators can be combined in a window comparator.
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wake-up from Sleep and Stop modes (through the EXTI controller)

## 15.3 COMP functional description

### 15.3.1 COMP block diagram

The block diagram of the comparators is shown in [Figure 51: Comparator 1 and 2 block diagrams](#).

Figure 51. Comparator 1 and 2 block diagrams



### 15.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the datasheet.

The output can also be internally redirected to a variety of timer input for the following purposes:

- Emergency shut-down of PWM signals, using BKIN
- Cycle-by-cycle current control, using OCREF\_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

### 15.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the PCLK (APB1 clock).

There is no clock enable control bit provided in the RCC controller.

*Note: **Important:** The polarity selection logic and the output redirection to the port works independently from the PCLK clock. This allows the comparator to work even in Stop mode.*

### 15.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

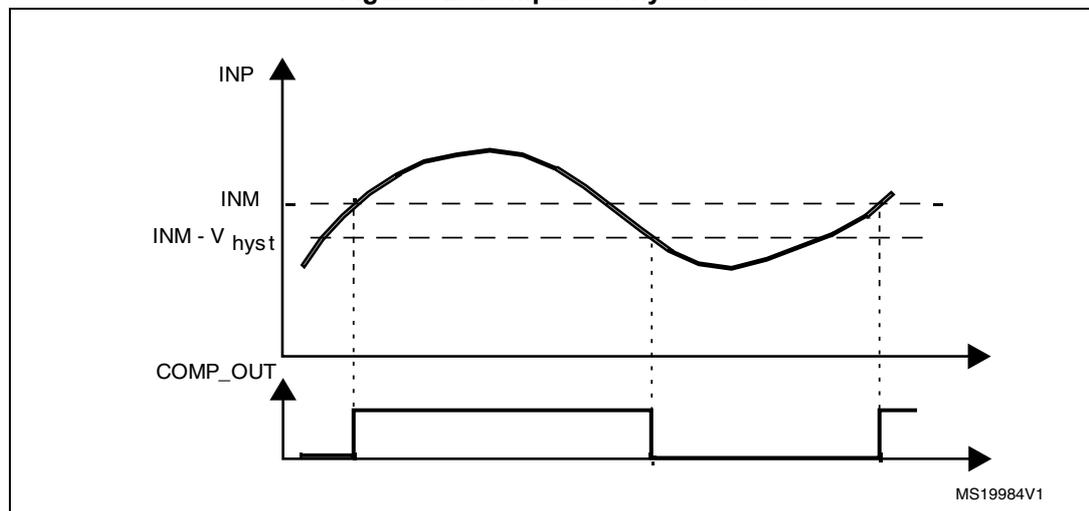
Once the programming is completed, using bits 30:16 and 15:0 of COMP\_CSR, the COMPxLOCK bit can be set to 1. This causes the whole COMP\_CSR register to become read-only, including the COMPxLOCK bit.

The write protection can only be reset by a MCU reset.

### 15.3.5 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

Figure 52. Comparator hysteresis



## 15.4 Power mode

The comparator power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application. The bits COMPxMODE[1:0] in COMP\_CSR register can be programmed as follows:

- 00: High speed / full power
- 01: Medium speed / medium power
- 10: Low speed / low-power
- 11: Very-low speed / ultra-low-power

## 15.5 Interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to Interrupt and events section for more details.

## 15.6 COMP registers

### 15.6.1 COMP control and status register (COMP\_CSR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP2LOCK	COMP2OUT	COMP2HYST [1:0]		COMP2POL	COMP2OUTSEL[2:0]			WNDWEN	COMP2INSEL[2:0]			COMP2MODE [1:0]		Res.	COMP2EN
rwo	r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r		rw/r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP1LOCK	COMP1OUT	COMP1HYST [1:0]		COMP1POL	COMP1OUTSEL[2:0]			Res.	COMP1INSEL[2:0]			COMP1MODE [1:0]		COMP1_INP_D AC	COMP1EN
rwo	r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r		rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r

Bit 31 **COMP2LOCK**: Comparator 2 lock

This bit is write-once. It is set by software. It can only be cleared by a system reset.

It allows to have all control bits of comparator 2 as read-only.

0: COMP\_CSR[31:16] bits are read-write.

1: COMP\_CSR[31:16] bits are read-only.

Bit 30 **COMP2OUT**: Comparator 2 output

This read-only bit is a copy of comparator 2 output state.

0: Output is low (non-inverting input below inverting input).

1: Output is high (non-inverting input above inverting input).

Bits 29:28 **COMP2HYST[1:0]** Comparator 2 hysteresis

These bits control the hysteresis level.

00: No hysteresis

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Please refer to the electrical characteristics for the hysteresis values.

Bit 27 **COMP2POL**: Comparator 2 output polarity

This bit is used to invert the comparator 2 output.

0: Output is not inverted

1: Output is inverted

Bits 26:24 **COMP2OUTSEL[2:0]**: Comparator 2 output selection

These bits select the destination of the comparator output.

000: No selection

001: Timer 16 break input

010: Timer 4 Input capture 1

011: Timer 4 OCrefclear input

100: Timer 2 input capture 4

101: Timer 2 OCrefclear input

110: Timer 3 input capture 1

111: Timer 3 OCrefclear input

- Bit 23 **WNDWEN**: Window mode enable  
This bit connects the non-inverting input of COMP2 to COMP1's non-inverting input, which is simultaneously disconnected from PA3.  
0: Window mode disabled  
1: Window mode enabled
- Bits 22:20 **COMP2INSEL[2:0]**: Comparator 2 inverting input selection  
These bits allows to select the source connected to the inverting input of the comparator 2.  
000: 1/4 of Vrefint  
001: 1/2 of Vrefint  
010: 3/4 of Vrefint  
011: Vrefint  
100: DAC1\_OUT1 output (and PA4 output)  
101: Input from PA5 (and DAC1\_OUT2 output)  
110: Input from PA2  
111: Input from PA6 (and DAC2\_OUT1 output)
- Bits 19:18 **COMP2MODE[1:0]**: Comparator 2 mode  
These bits control the operating mode of the comparator2 and allows to adjust the speed/consumption.  
00: High speed/  
01: Medium speed  
10: Low-power  
11: Ultra-low-power
- Bit 17 Reserved, must be kept at reset value.
- Bit 16 **COMP2EN**: Comparator 2 enable  
This bit switches ON/OFF the comparator2.  
0: Comparator 2 disabled  
1: Comparator 2 enabled
- Bit 15 **COMP1LOCK**: Comparator 1 lock  
This bit is write-once. It is set by software. It can only be cleared by a system reset.  
It allows to have all control bits of comparator 1 as read-only.  
0: COMP\_CSR[15:0] bits are read-write.  
1: COMP\_CSR[15:0] bits are read-only.
- Bit 14 **COMP1OUT**: Comparator 1 output  
This read-only bit is a copy of comparator 1 output state.  
0: Output is low (non-inverting input below inverting input).  
1: Output is high (non-inverting input above inverting input).
- Bits 13:12 **COMP1HYST[1:0]** Comparator 1 hysteresis  
These bits are controlling the hysteresis level.  
00: No hysteresis  
01: Low hysteresis  
10: Medium hysteresis  
11: High hysteresis  
Please refer to the electrical characteristics for the hysteresis values.
- Bit 11 **COMP1POL**: Comparator 1 output polarity  
This bit is used to invert the comparator 1 output.  
0: output is not inverted  
1: output is inverted

Bits 10:8 **COMP1OUTSEL[2:0]**: Comparator 1 output selection

These bits selects the destination of the comparator 1 output.

- 000: no selection
- 001: Timer 15 break input
- 010: Timer 3 Input capture 1
- 011: Timer 3 OCrefclear input
- 100: Timer 2 input capture 4
- 101: Timer 2 OCrefclear input
- 110: Timer 5 input capture 4
- 111: Timer 5 OCrefclear input

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **COMP1INSEL[2:0]**: Comparator 1 inverting input selection

These bits select the source connected to the inverting input of the comparator 1.

- 000: 1/4 of Vrefint
- 001: 1/2 of Vrefint
- 010: 3/4 of Vrefint
- 011: Vrefint
- 100: DAC1\_OUT1output (and PA4)
- 101: Input from PA5 (and DAC1\_OUT2 output)
- 110: Input from PA0
- 111: Input from PA6 (and DAC2\_OUT1 output)

Bits 3:2 **COMP1MODE[1:0]**: Comparator 1 mode

These bits control the operating mode of the comparator1 and allows to adjust the speed/consumption.

- 00: Ultra-low-power
- 01: Low-power
- 10: Medium speed
- 11: High speed

Bit 1 **COMP1\_INP\_DAC**: Comparator 1 non-inverting input connection to DAC output.

This bit closes a switch between comparator 1 non-inverting input on PA0 and PA4 (DAC) I/O.

- 0: Switch open
- 1: Switch closed

*Note: This switch is solely intended to redirect signals onto high impedance input, such as COMP1 non-inverting input (highly resistive switch).*

Bit 0 **COMP1EN**: Comparator 1 enable

This bit switches COMP1 ON/OFF.

- 0: Comparator 1 disabled
- 1: Comparator 1 enabled

### 15.6.2 COMP register map

The following table summarizes the comparator registers.

**Table 43. COMP 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
0x1C	COMP_CSR	COMP2LOCK	COMP2OUT	COMP2HYST[1:0]	COMP2POL	COMP2OUTSEL[2:0]	COMP2OUTSEL[2:0]	COMP2OUTSEL[2:0]	WINDWEN	COMP2INSEL[2:0]	COMP2INSEL[2:0]	COMP2INSEL[2:0]	COMP2MODE[1:0]	Res.	COMP2EN	COMP1LOCK	COMP1OUT	COMP1HYST[1:0]	COMP1POL	COMP1OUTSEL[2:0]	COMP1OUTSEL[2:0]	Res.	COMP1INSEL[2:0]	COMP1INSEL[2:0]	COMP1MODE[1:0]	COMP1INP_DAC	COMP1EN						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2](#) for the register boundary addresses.



## 16 General-purpose timers (TIM2 to TIM5, TIM19)

### 16.1 TIM2 to TIM5/TIM19 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

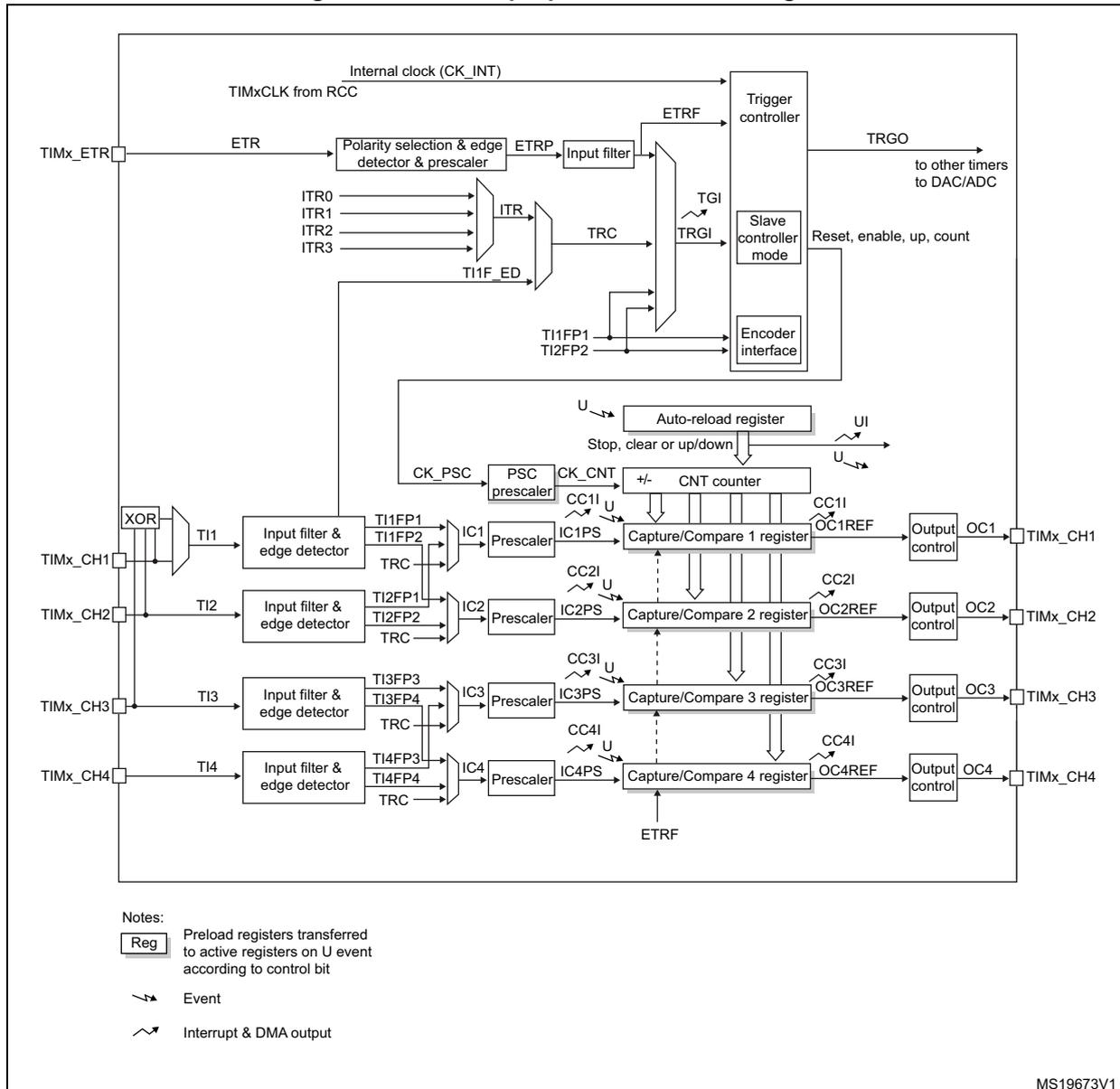
The timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 16.3.15](#).

### 16.2 TIM2 to TIM5/TIM19 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4, and TIM19) or 32-bit (TIM2 and TIM5) 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 53. General-purpose timer block diagram



## 16.3 TIM2 to TIM5/TIM19 functional description

### 16.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC):
- Auto-Reload Register (TIMx\_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT\_EN is set 1 clock cycle after CEN.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 54 and Figure 16.3.2 give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 54. Counter timing diagram with prescaler division change from 1 to 2**

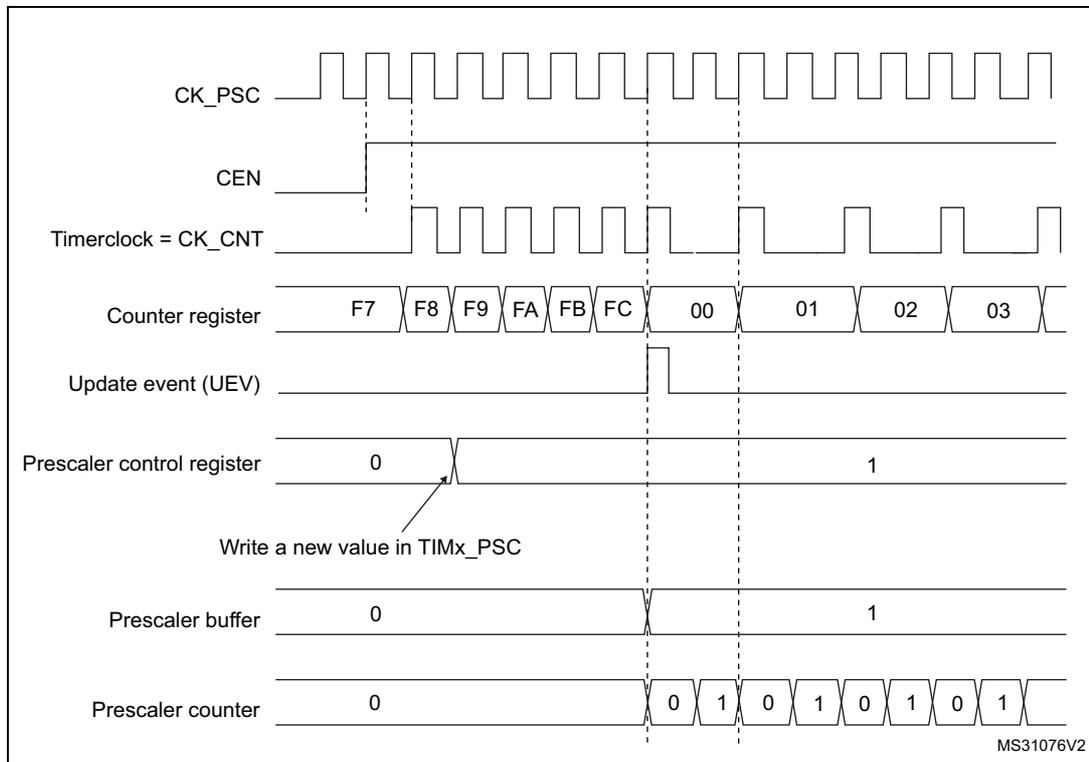
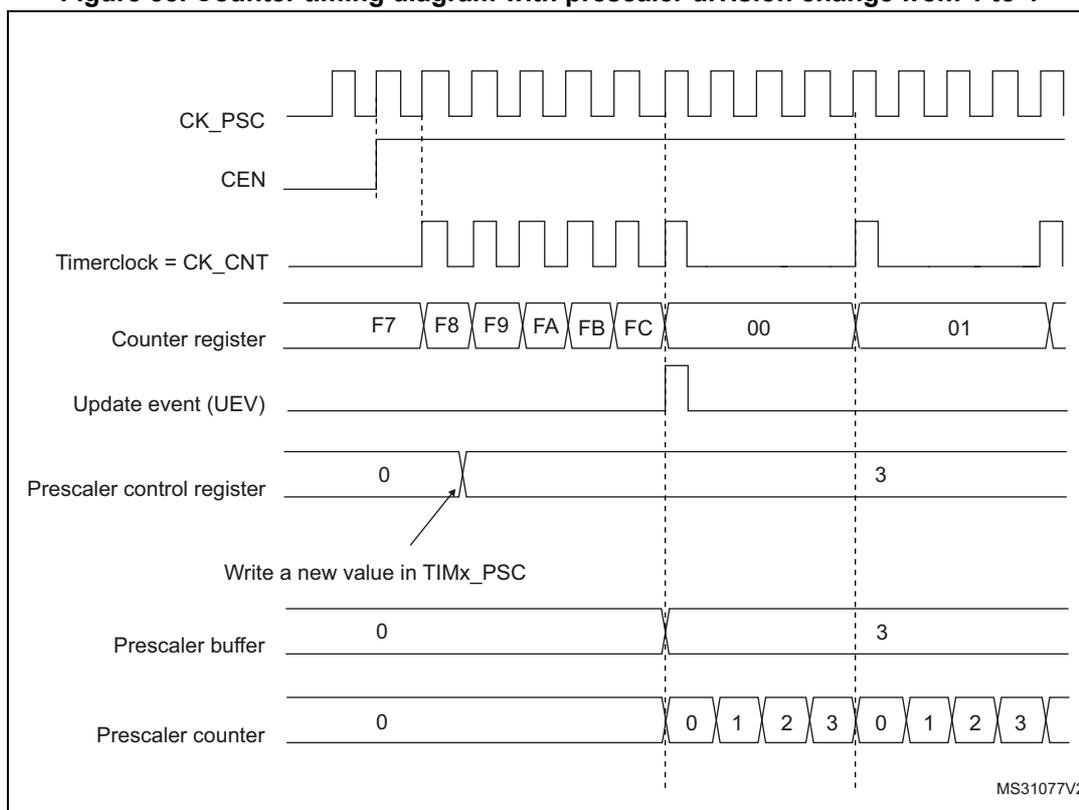


Figure 55. Counter timing diagram with prescaler division change from 1 to 4



### 16.3.2 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 56. Counter timing diagram, internal clock divided by 1

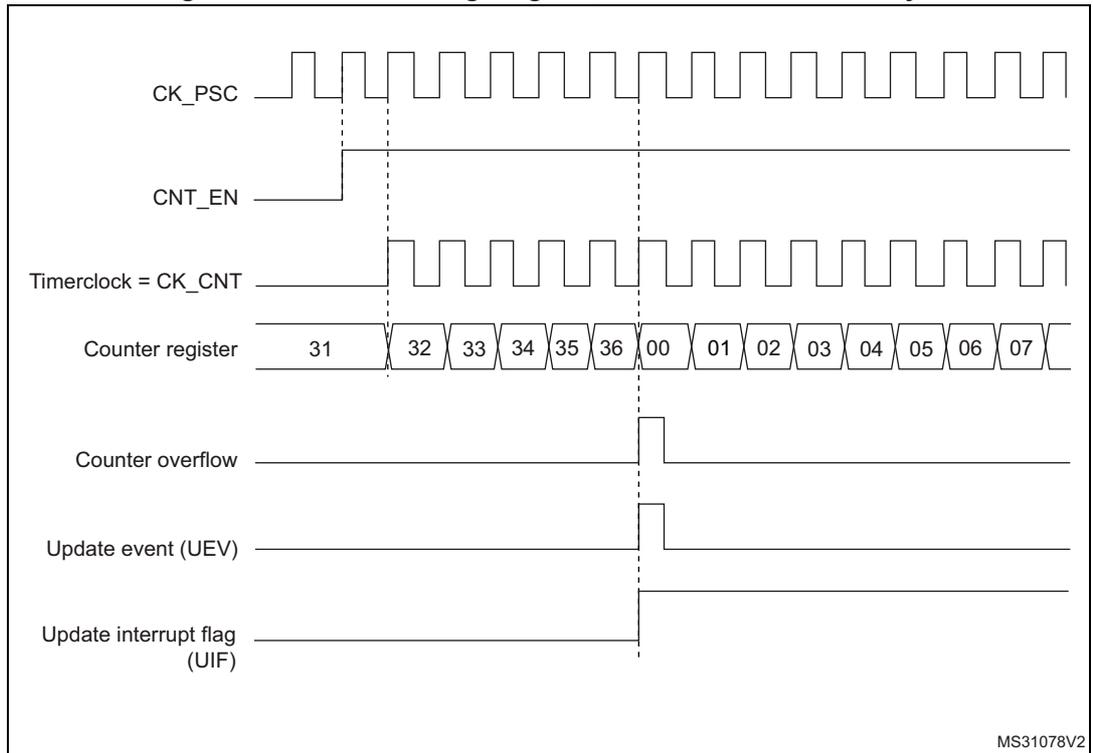


Figure 57. Counter timing diagram, internal clock divided by 2

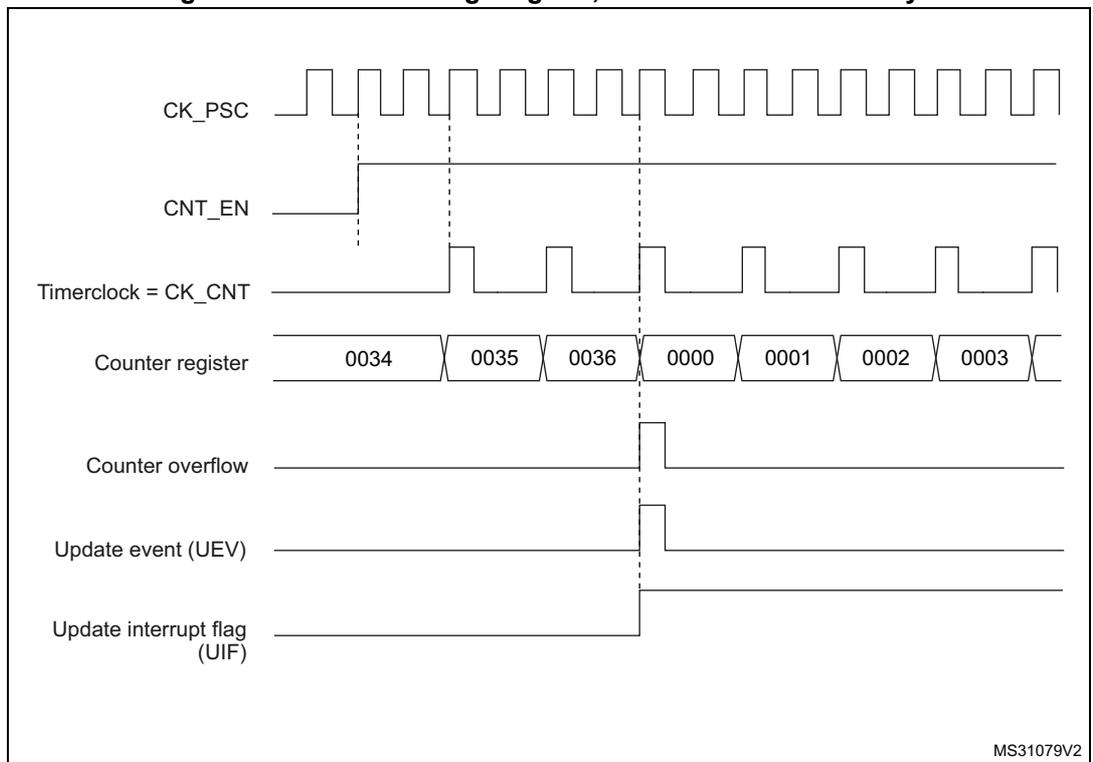


Figure 58. Counter timing diagram, internal clock divided by 4

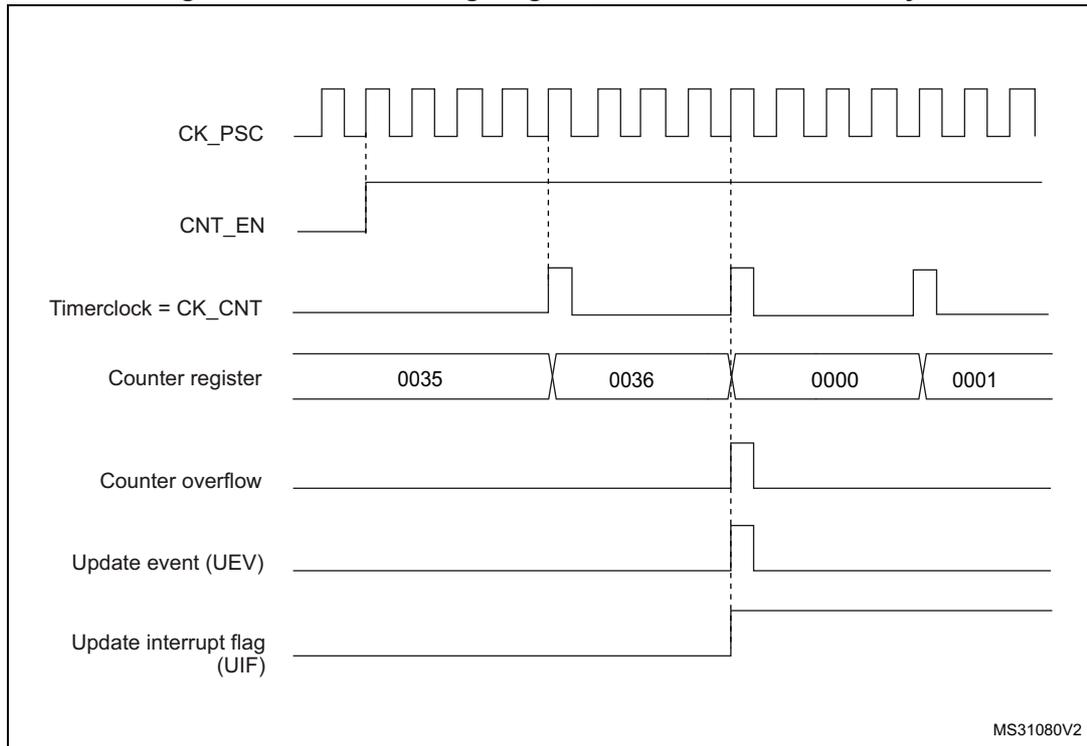
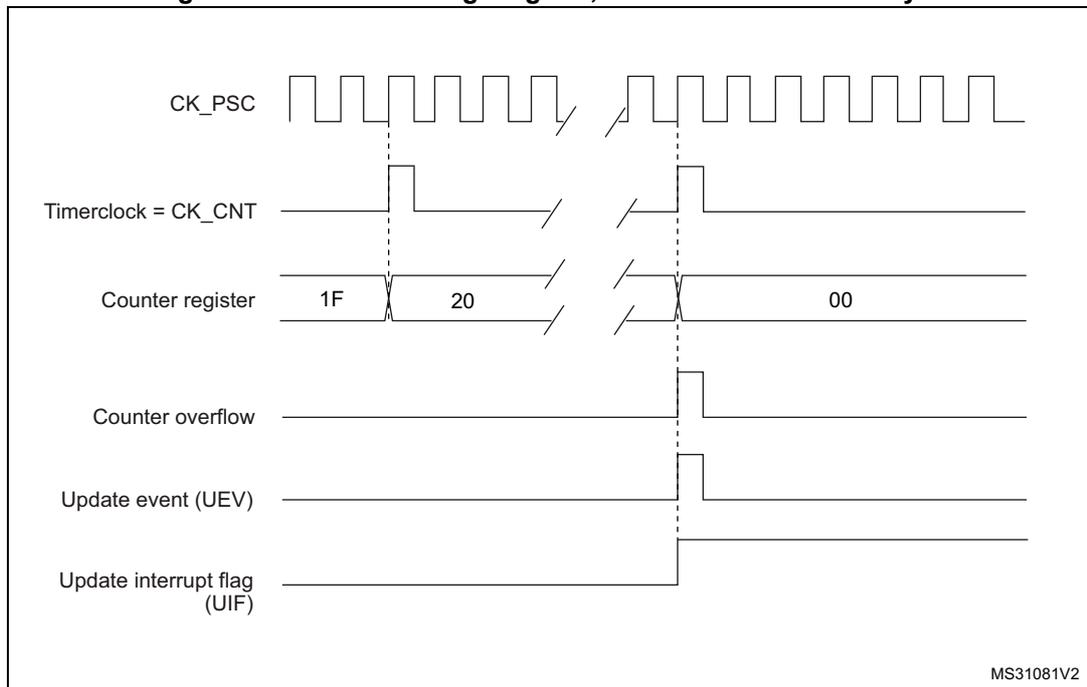
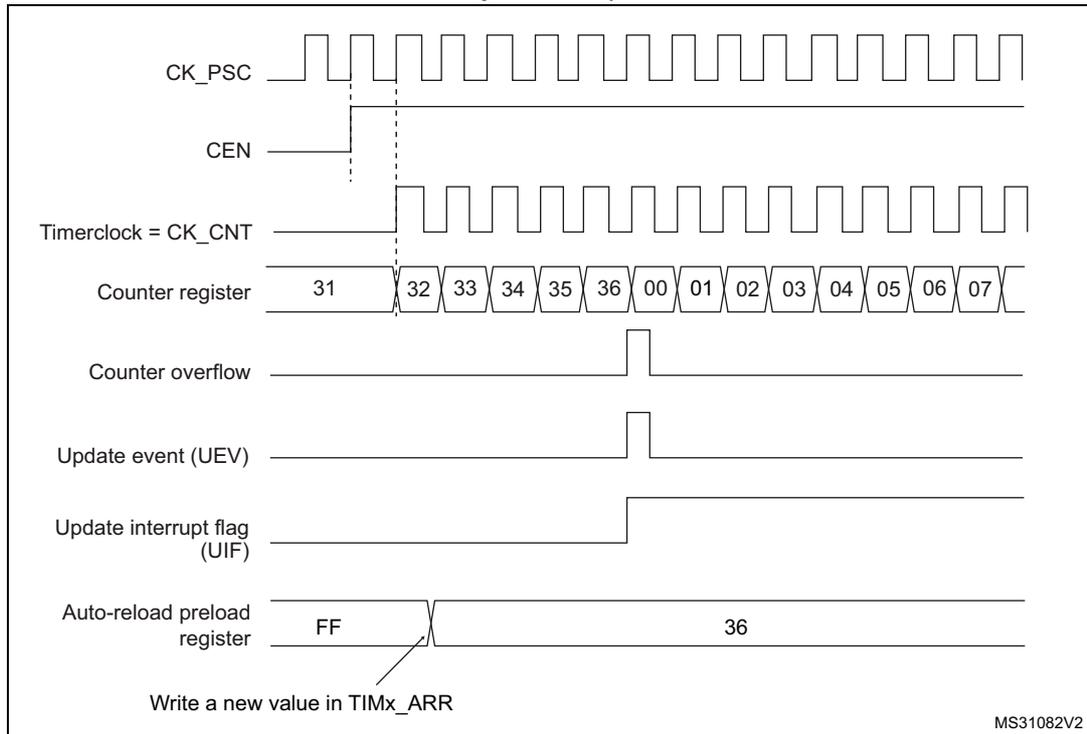


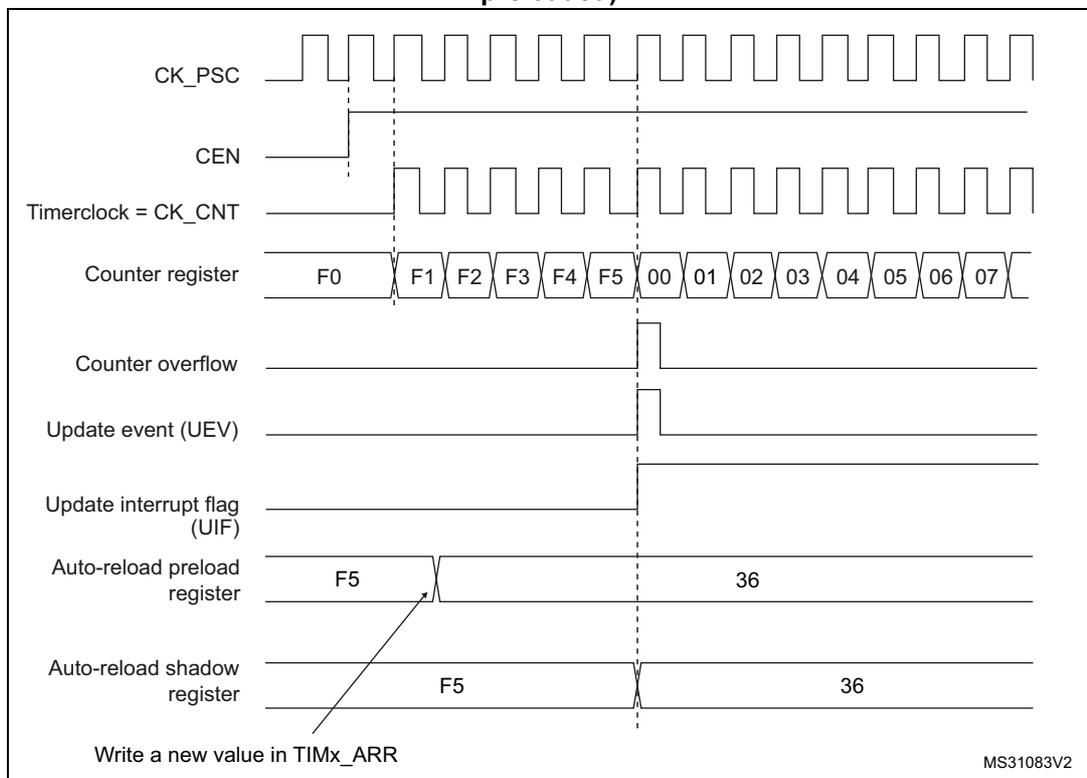
Figure 59. Counter timing diagram, internal clock divided by N



**Figure 60. Counter timing diagram, Update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 61. 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 62. Counter timing diagram, internal clock divided by 1**

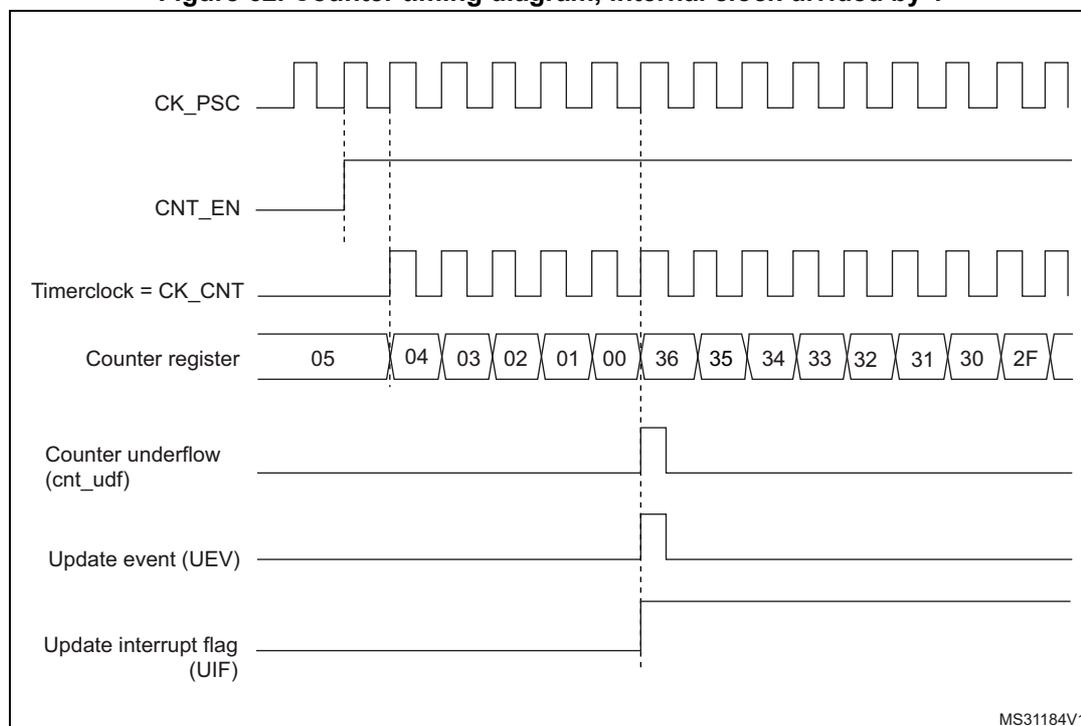


Figure 63. Counter timing diagram, internal clock divided by 2

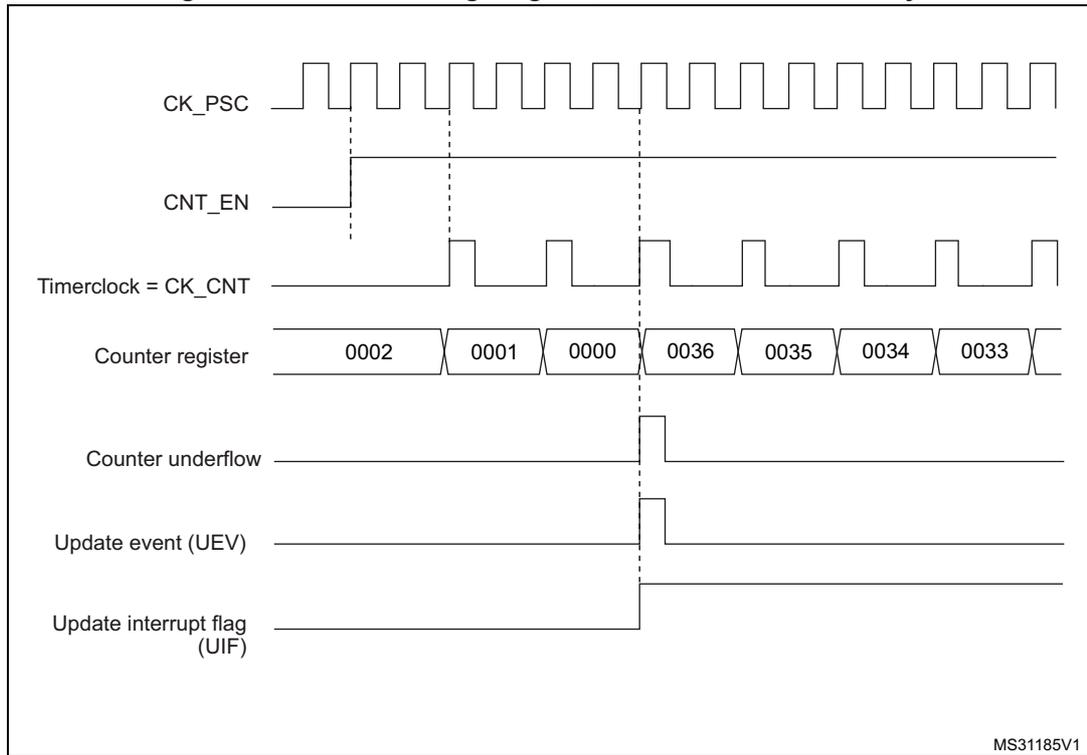
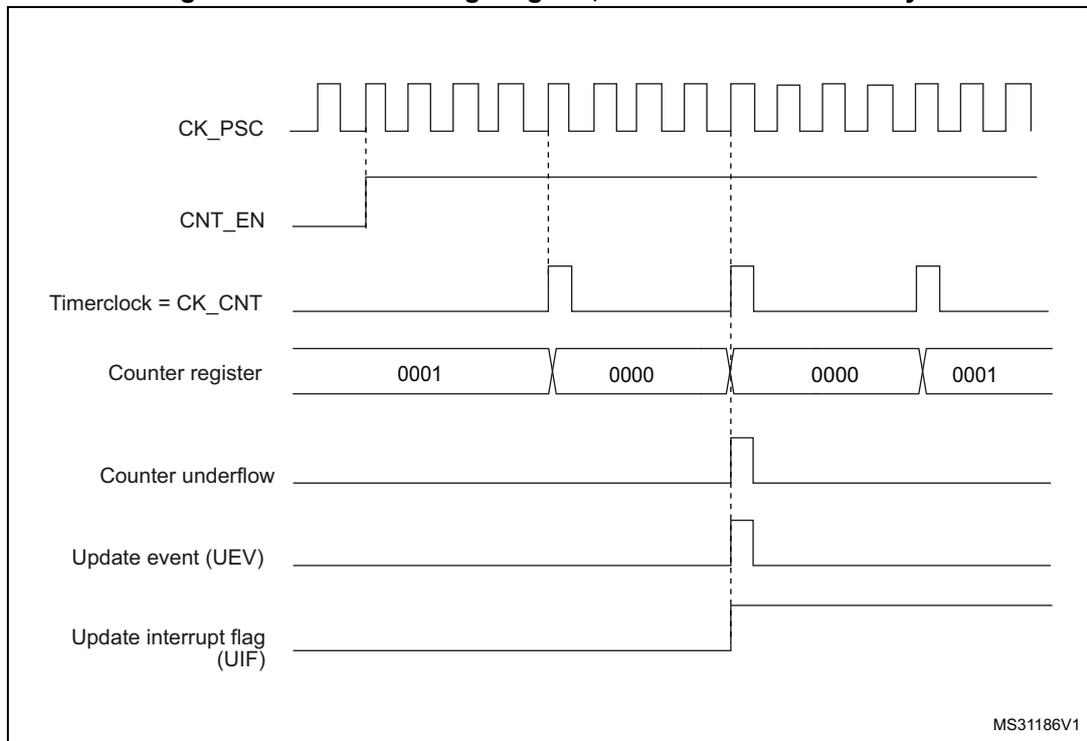
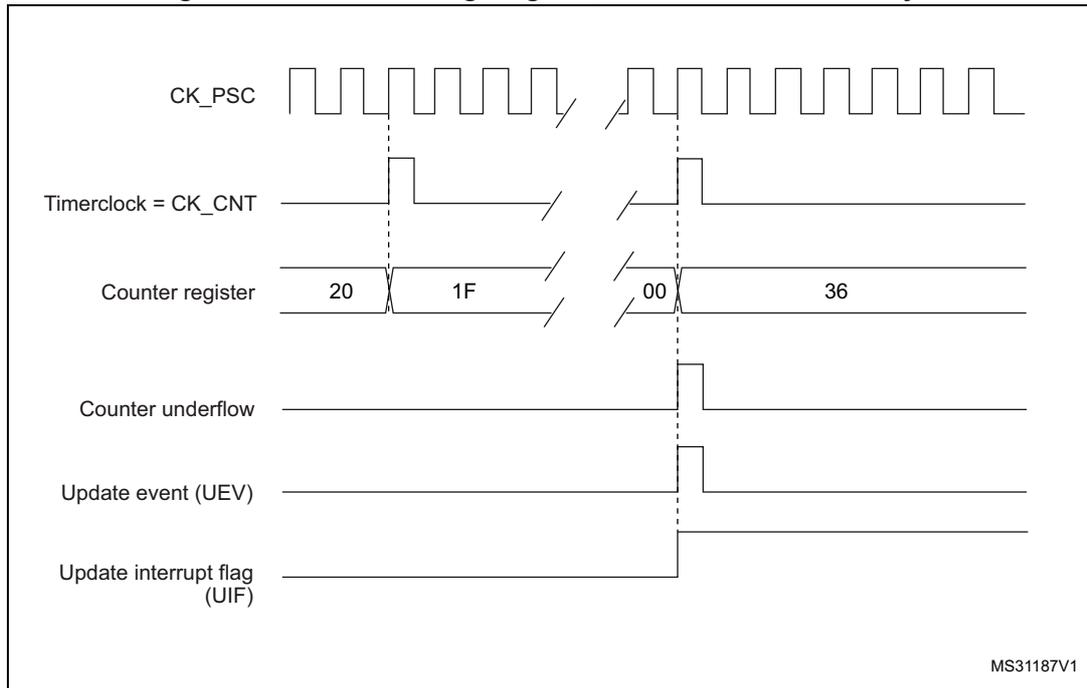


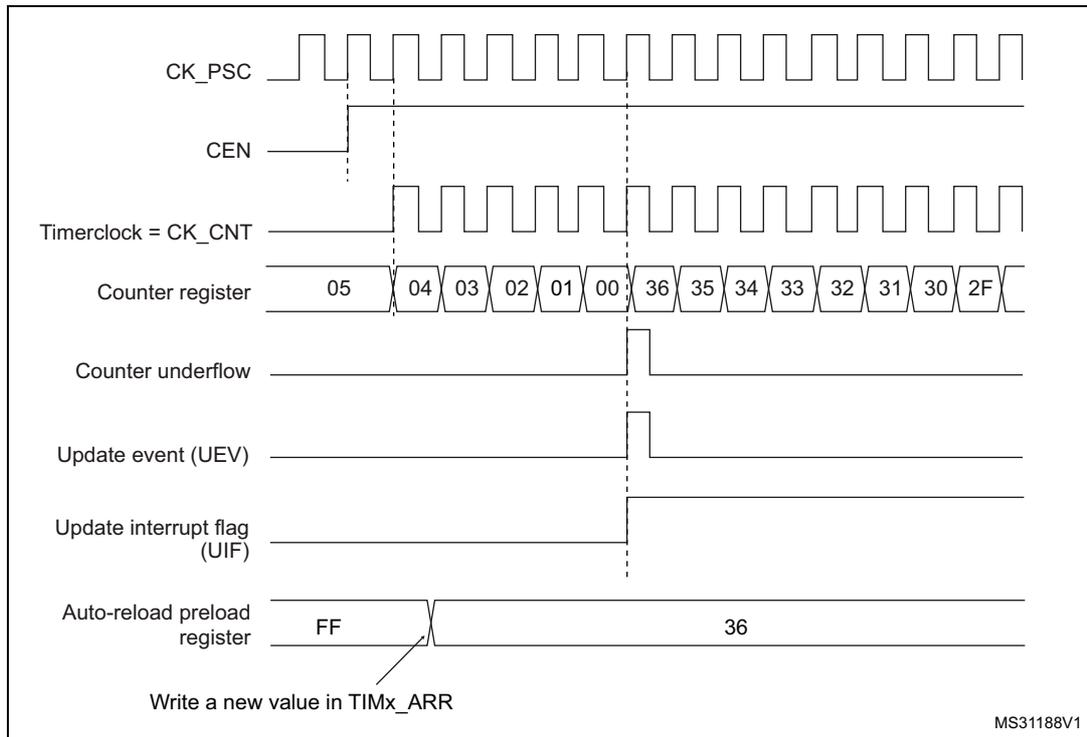
Figure 64. Counter timing diagram, internal clock divided by 4



**Figure 65. Counter timing diagram, internal clock divided by N**



**Figure 66. 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.

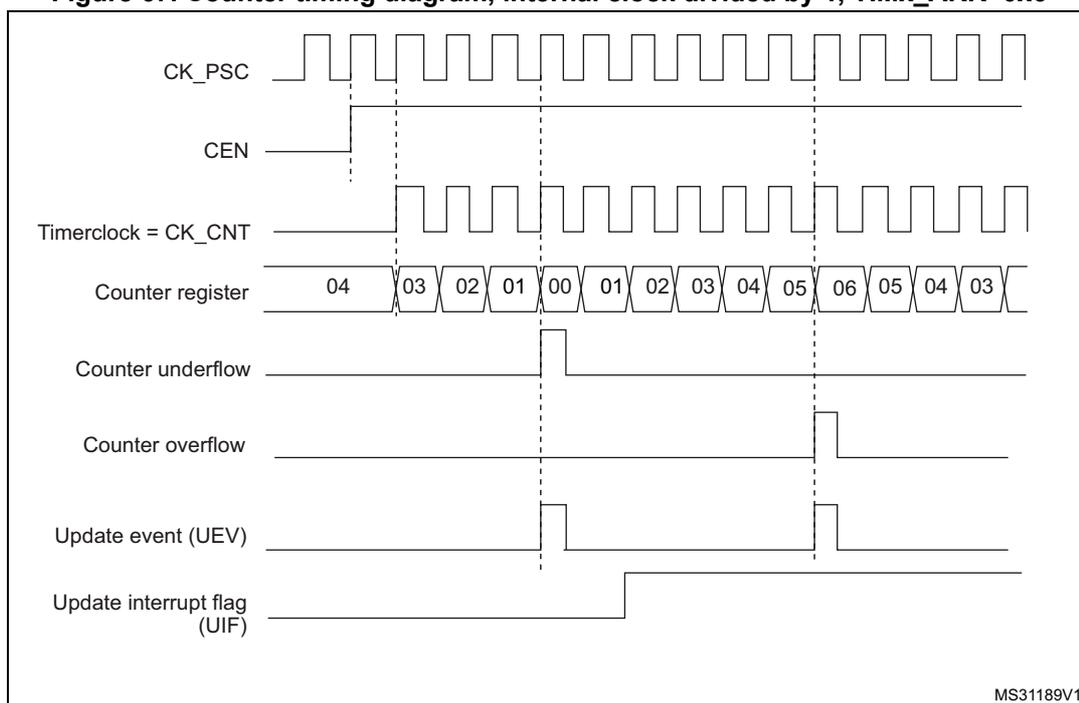
In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on 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 67. 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 16.4.1: TIMx control register 1 \(TIMx\\_CR1\)](#) on page 335).

**Figure 68. Counter timing diagram, internal clock divided by 2**

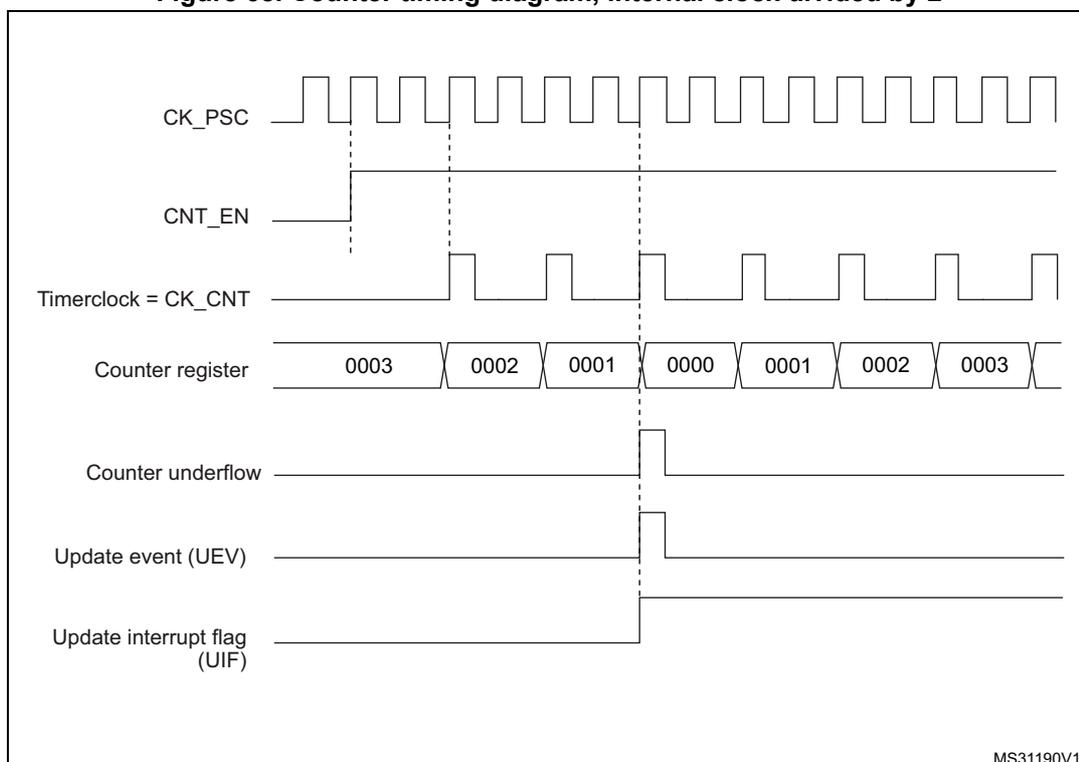
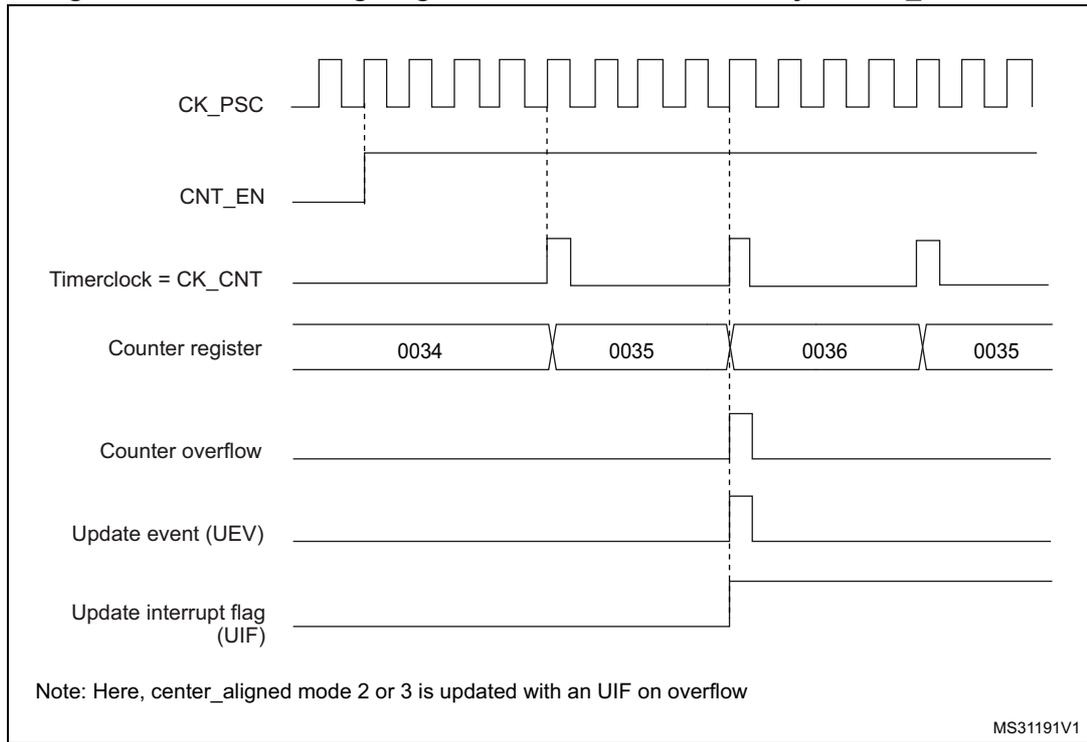
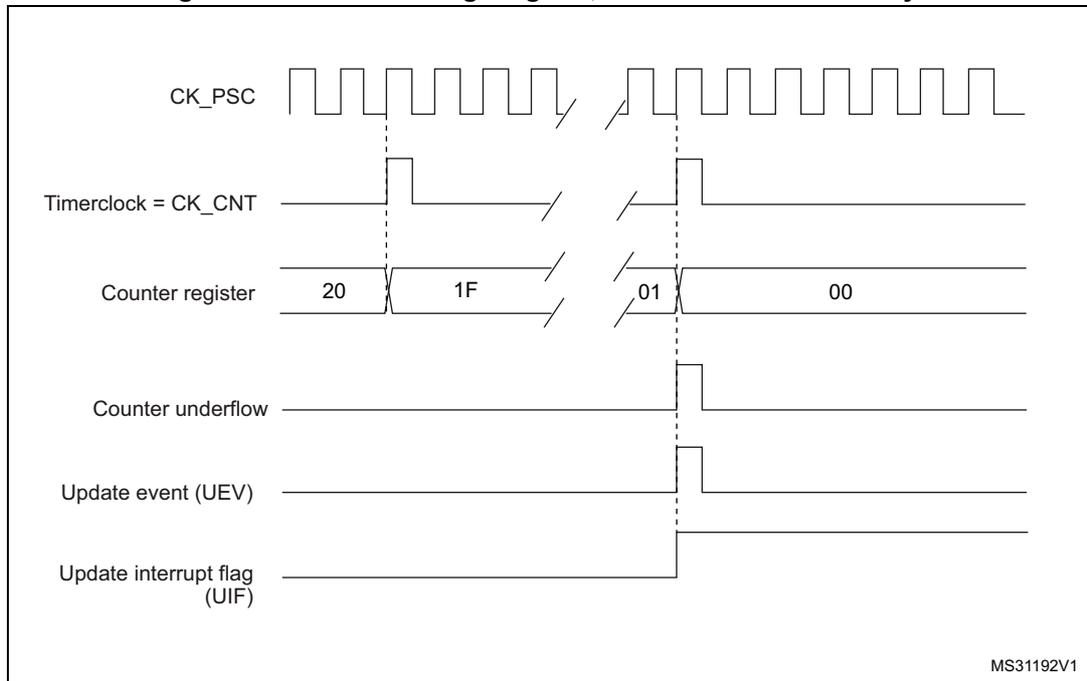


Figure 69. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36

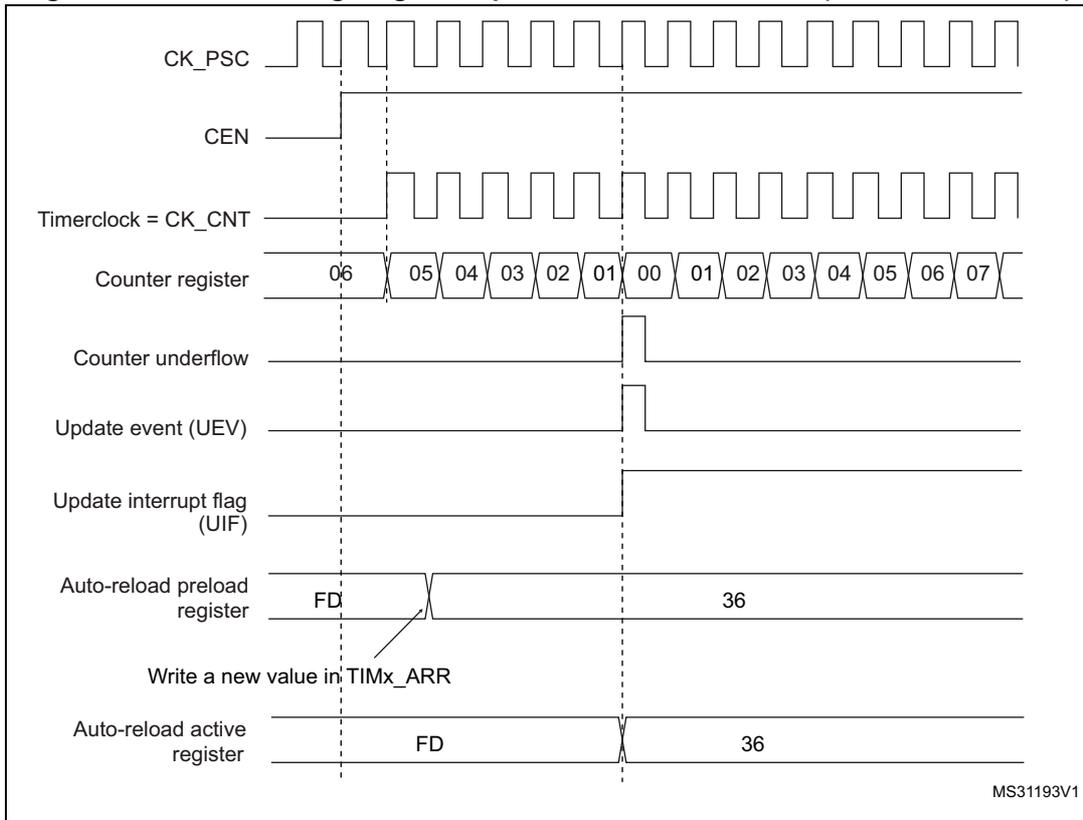


1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

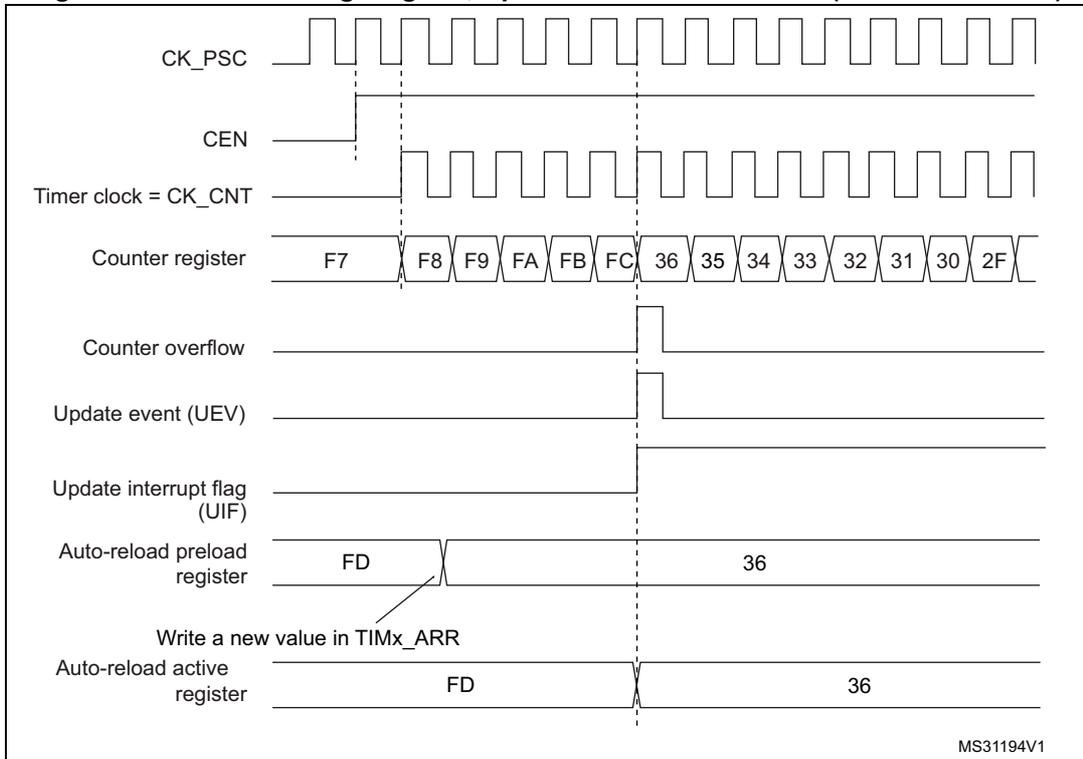
Figure 70. Counter timing diagram, internal clock divided by N



**Figure 71. Counter timing diagram, Update event with ARPE=1 (counter underflow)**



**Figure 72. Counter timing diagram, Update event with ARPE=1 (counter overflow)**



### 16.3.3 Clock selection

The counter clock can be provided by the following clock sources:

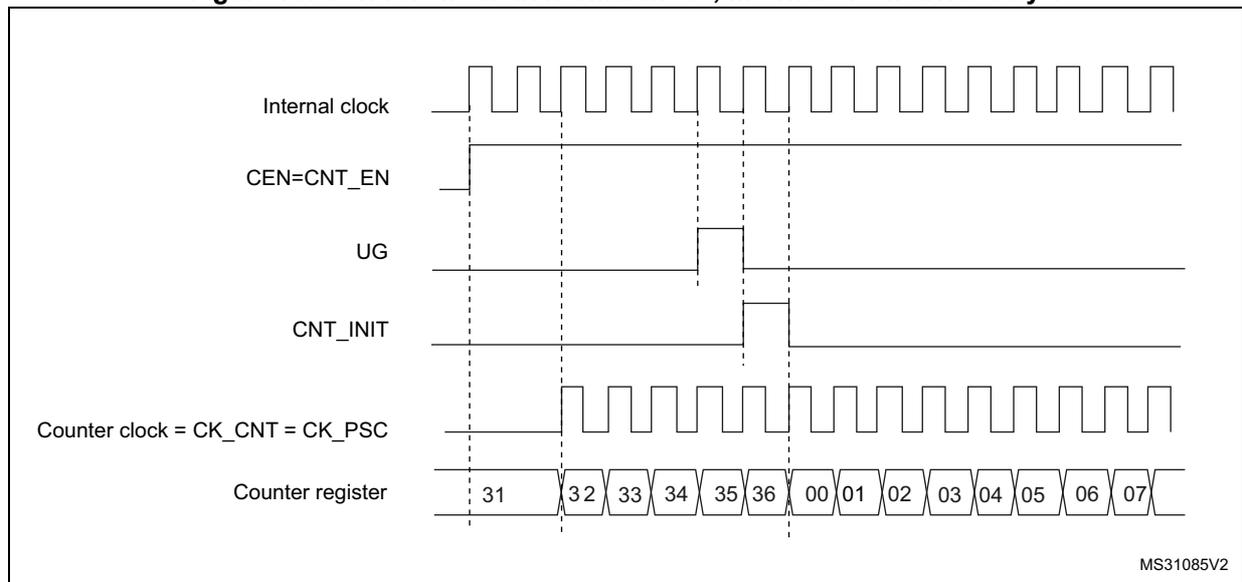
- Internal clock (CK\_INT)
- External clock mode1: external input pin (TlX)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer. Refer to : [Using one timer as prescaler for another on page 329](#) for more details.

#### 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 73](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

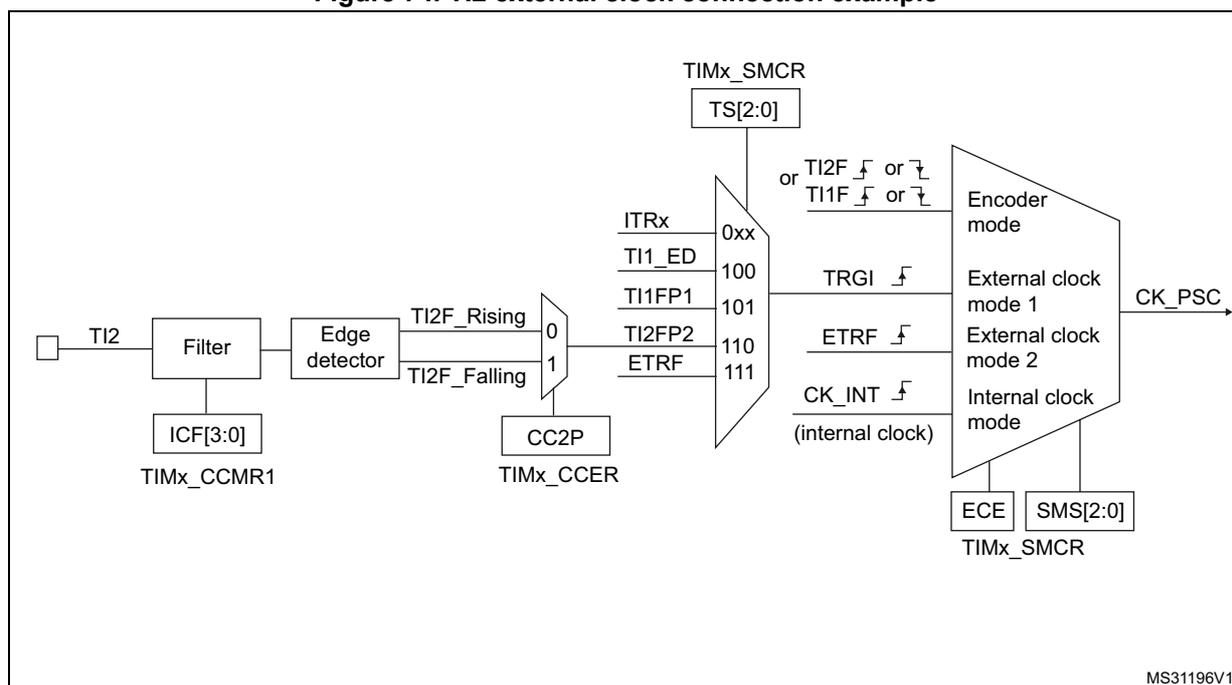
**Figure 73. Control circuit in normal mode, internal clock divided by 1**



#### 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 74. 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:

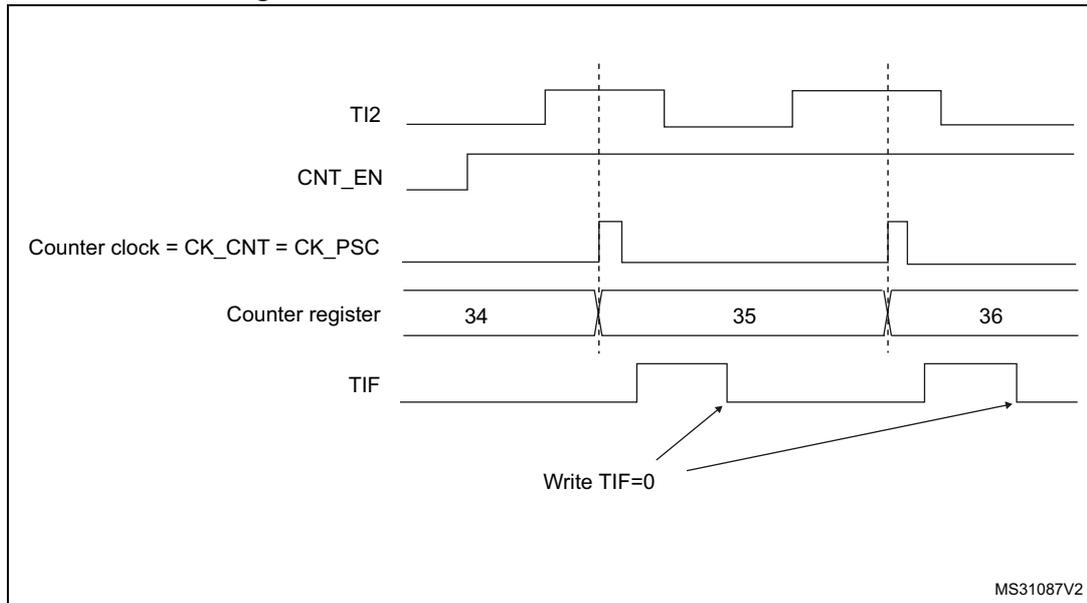
*The capture prescaler is not used for triggering, so you don't need to configure it.*

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 75. Control circuit in external clock mode 1



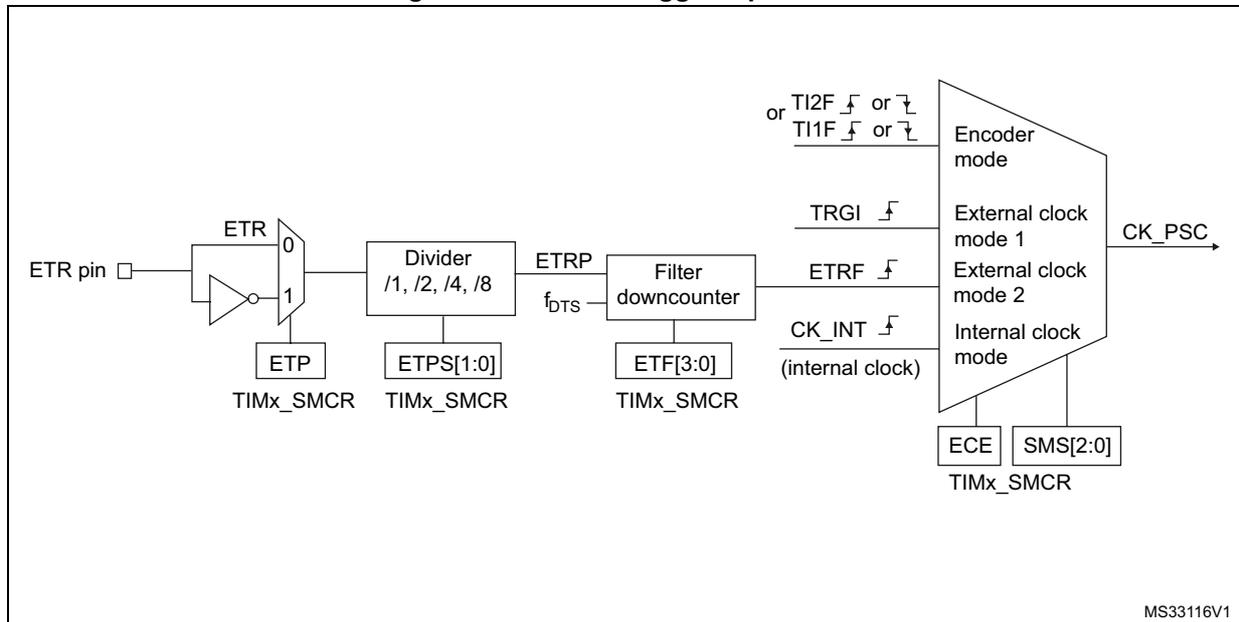
**External clock source mode 2**

This mode is selected by writing ECE=1 in the TIMx\_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Figure 76 gives an overview of the external trigger input block.

Figure 76. External trigger input block



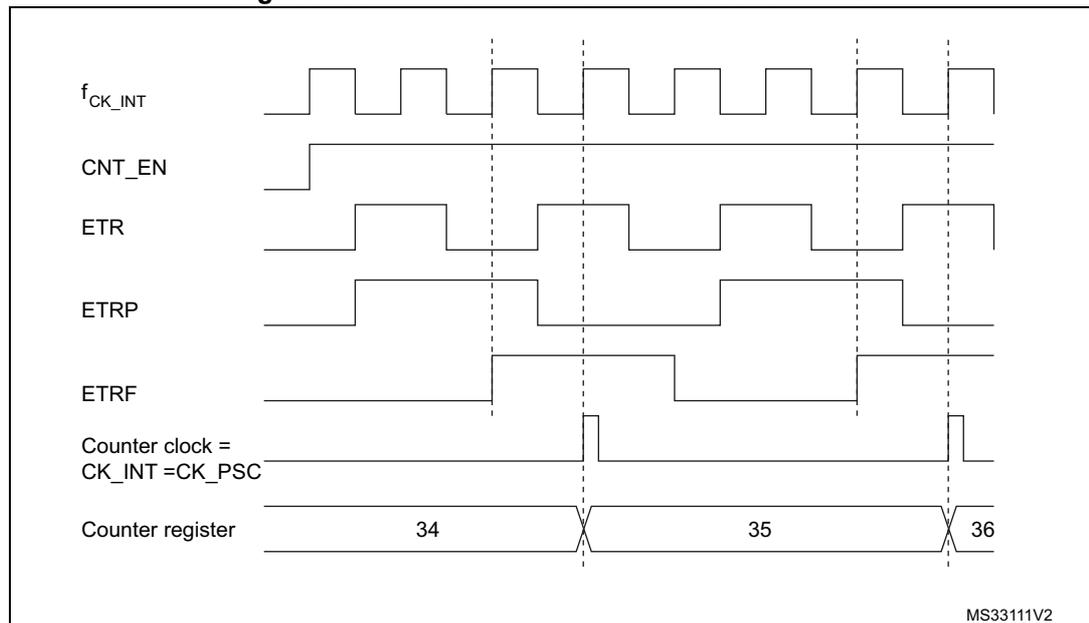
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

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 77. Control circuit in external clock mode 2**



### 16.3.4 Capture/compare channels

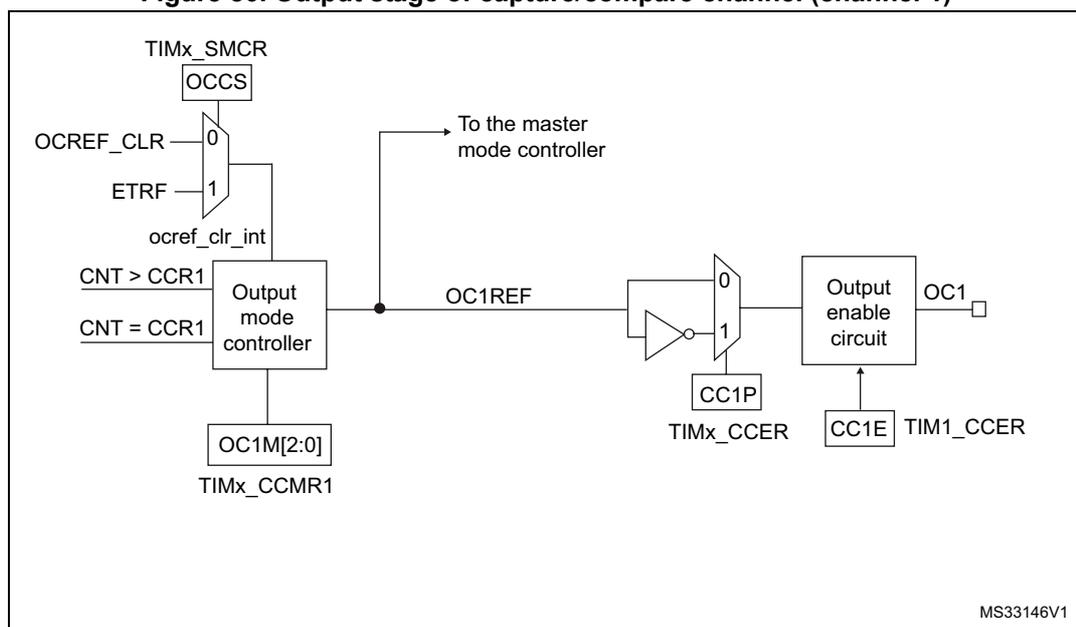
Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

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 80. 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.

### 16.3.5 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:

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 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 most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.

3. Select the edge of the active transition on the TI1 channel by writing the 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, 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.

### 16.3.6 PWM input mode

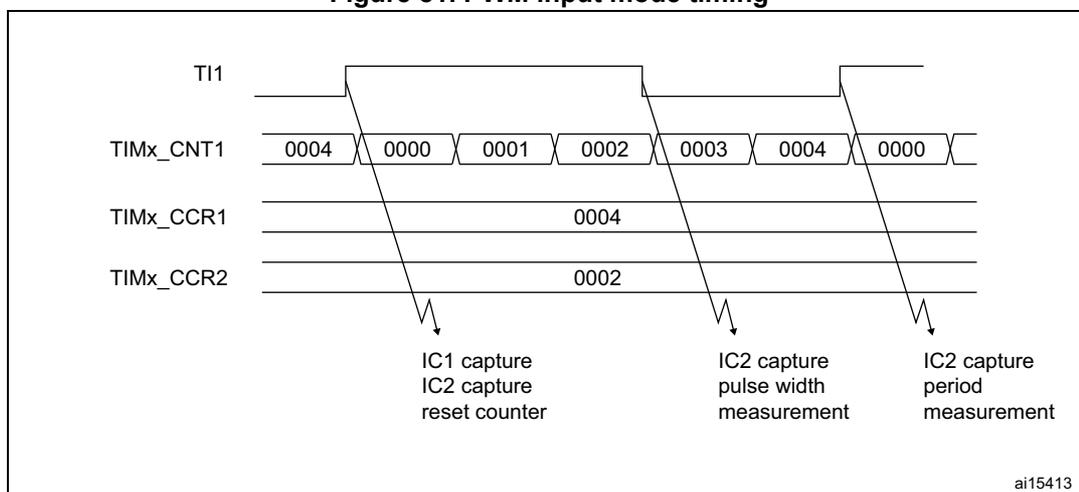
This mode is a particular case of input capture mode. The procedure is the same except:

- 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):

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): write the CC1P to '0' and the CC1NP bit to '0' (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): write the CC2P bit to '1' and the CC2NP bit to '0'(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 81. PWM input mode timing**



### 16.3.7 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.

### 16.3.8 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.

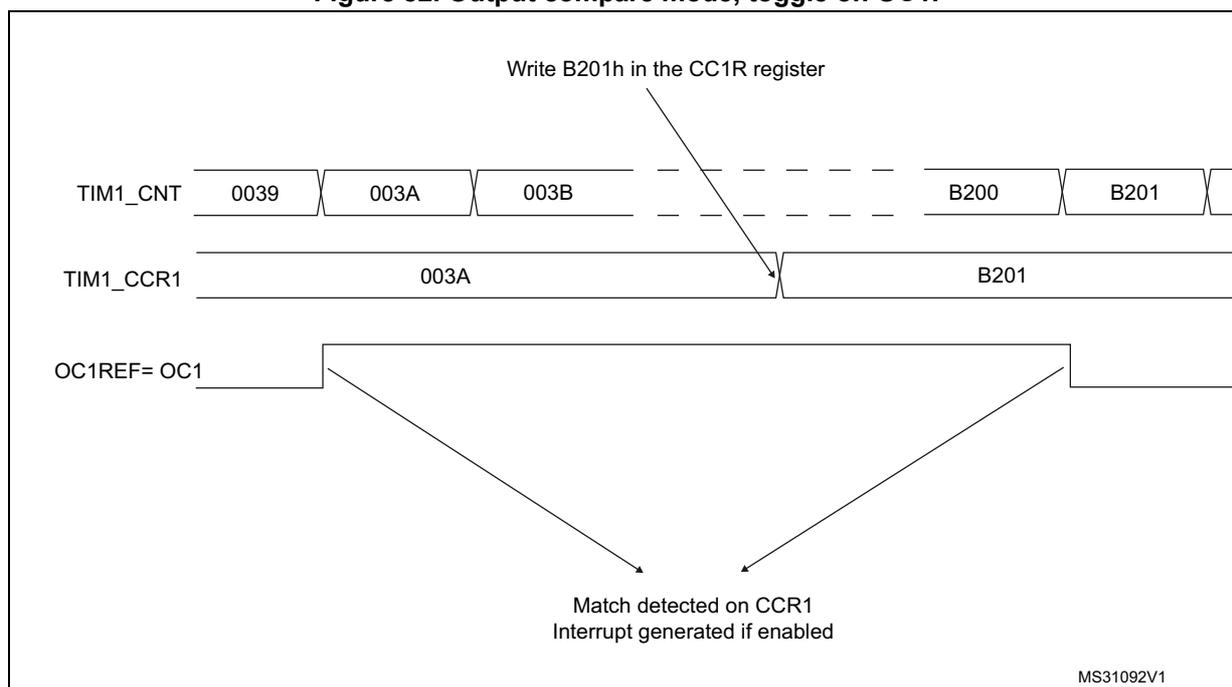
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 82](#).

Figure 82. Output compare mode, toggle on OC1.



### 16.3.9 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  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  (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.

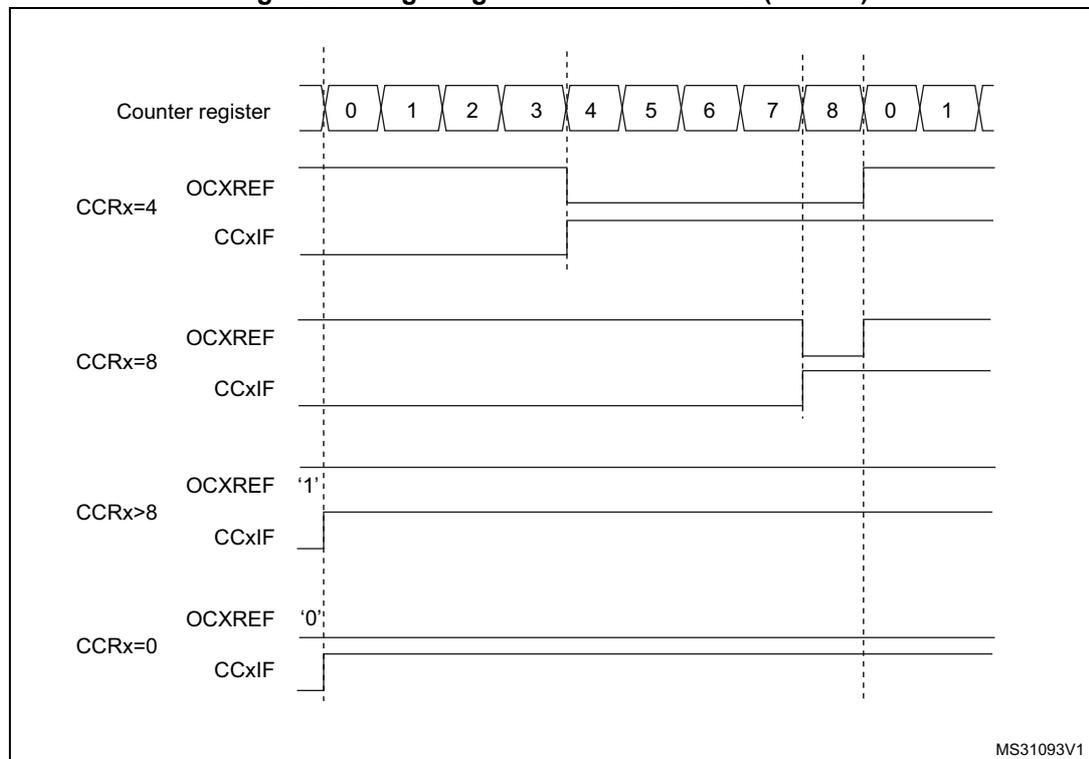
**PWM edge-aligned mode**

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to [Section : Upcounting mode on page 298](#).

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 83](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 83. Edge-aligned PWM waveforms (ARR=8)**



**Downcounting configuration**

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to [Section : Downcounting mode on page 302](#).

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**

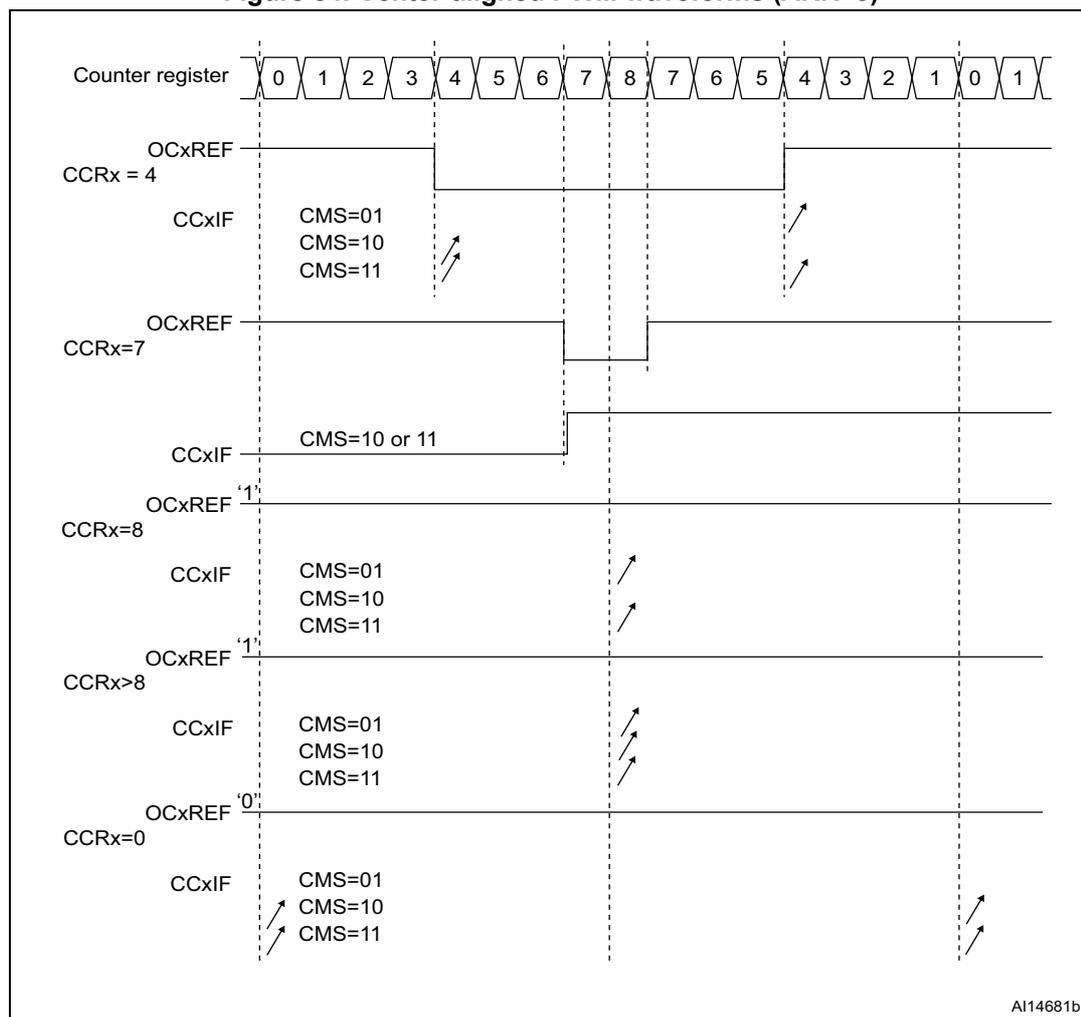
Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00 (all the remaining configurations having the same effect on the OCxREF/OCx signals).

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 [Section : Center-aligned mode \(up/down counting\) on page 305](#).

Figure 84 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 84. 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.

### 16.3.10 One-pulse mode

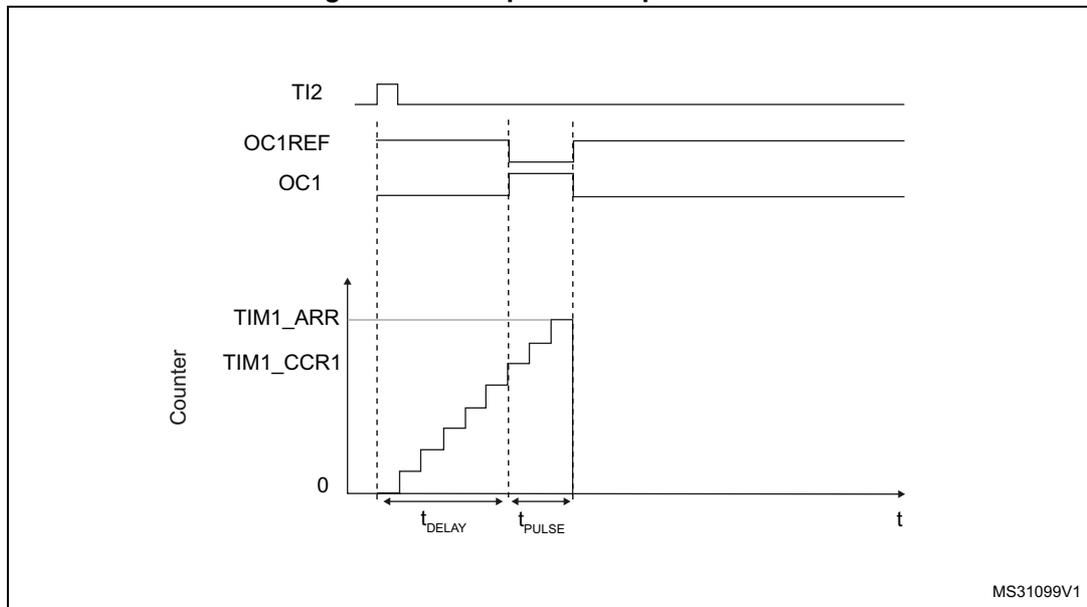
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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:  $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ ),
- In downcounting:  $CNT > CCRx$ .

Figure 85. Example of one-pulse mode.



MS31099V1

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 and CC2NP='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.

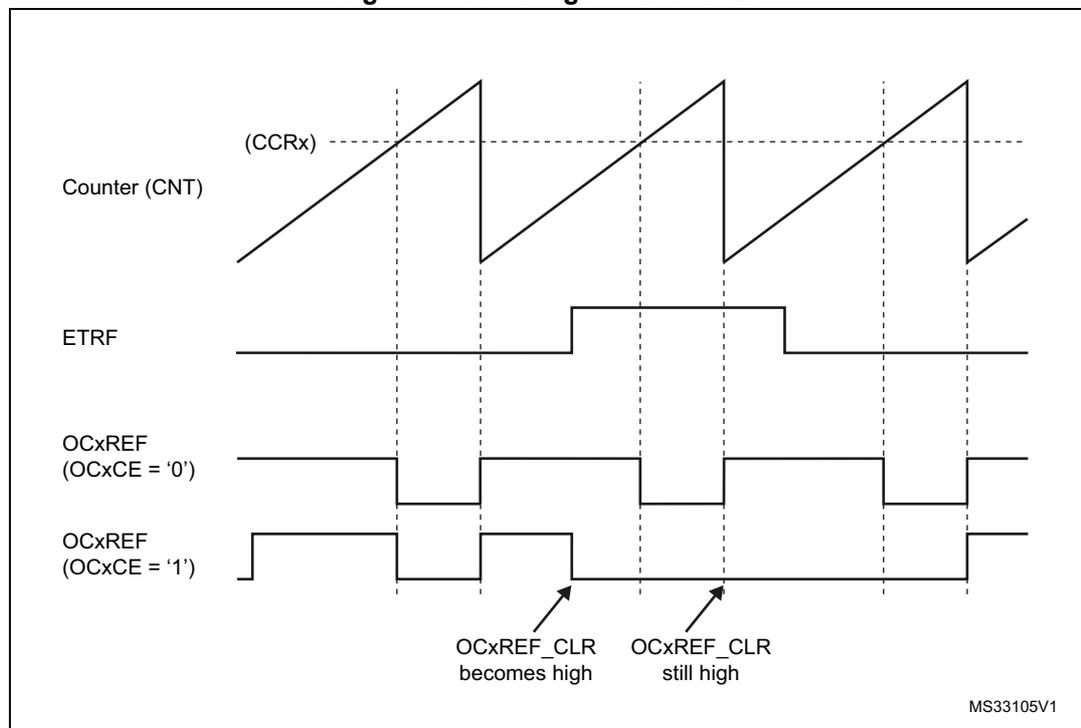
If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx\_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 16.3.11 Clearing the OCxREF signal on an external event

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 TIMx\_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 86 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 86. Clearing TIMx OCxREF



1. In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

### 16.3.12 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 44. 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 T11 and T12 don't switch at the same time.

**Table 44. Counting direction versus encoder signals**

Active edge	Level on opposite signal (TI1FP1 for T12, TI2FP2 for T11)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on T11 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on T12 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on T11 and T12	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 87 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 T11)
- CC2S= 01 (TIMx\_CCMR2 register, TI2FP2 mapped on T12)
- CC1P=0, CC1NP = '0' (TIMx\_CCER register, TI1FP1 noninverted, TI1FP1=T11)
- CC2P=0, CC2NP = '0' (TIMx\_CCER register, TI2FP2 noninverted, TI2FP2=T12)
- SMS= 011 (TIMx\_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx\_CR1 register, Counter is enabled)

**Figure 87. Example of counter operation in encoder interface mode**

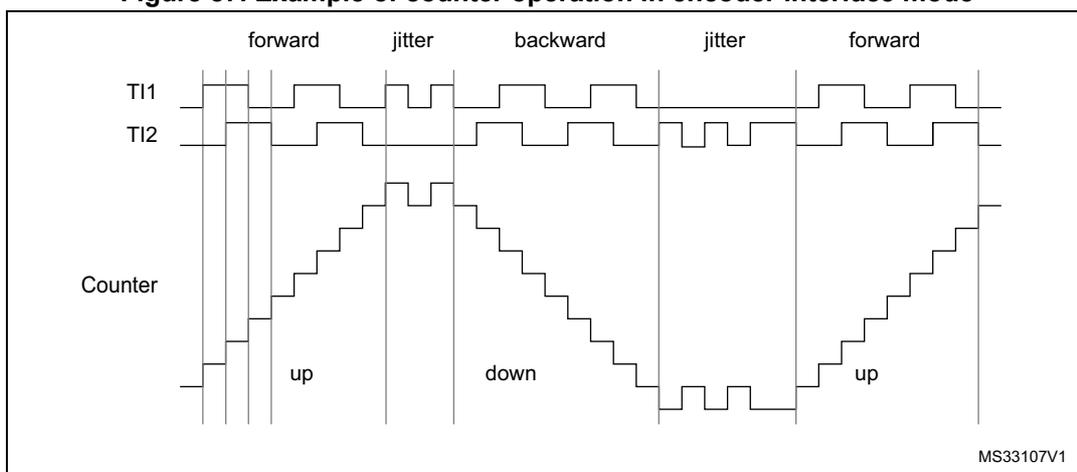
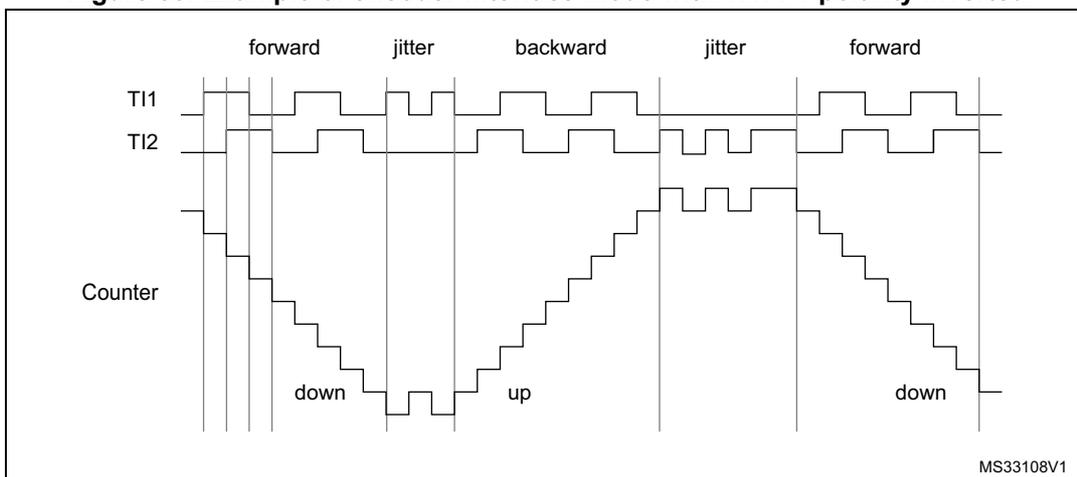


Figure 88 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

**Figure 88. 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.

### 16.3.13 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 to TIMx\_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

### 16.3.14 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.

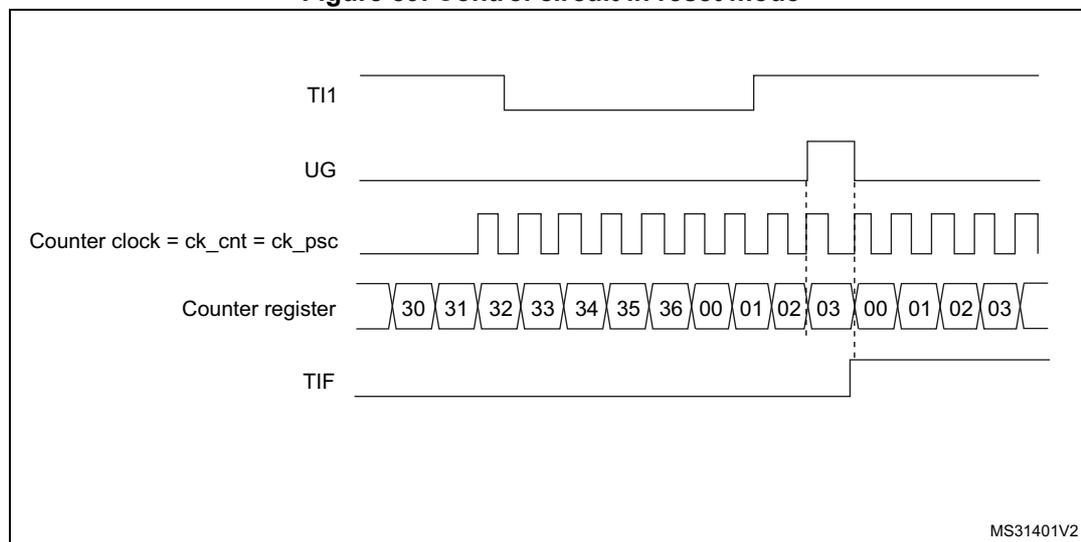
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- 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 89. Control circuit in reset mode



**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

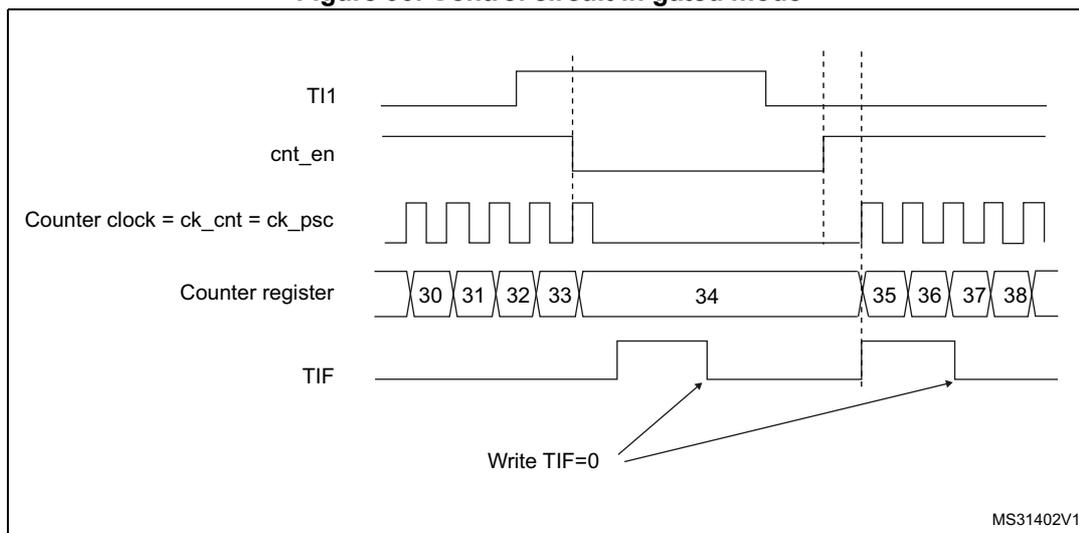
In the following example, the upcounter counts only when TI1 input is low:

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. Write 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 90. Control circuit in gated mode**



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we 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

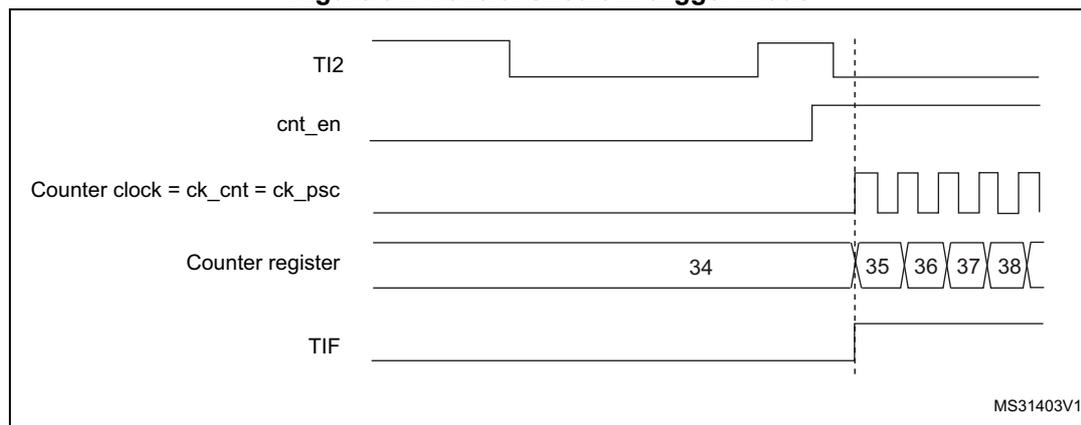
CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).

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 91. 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.

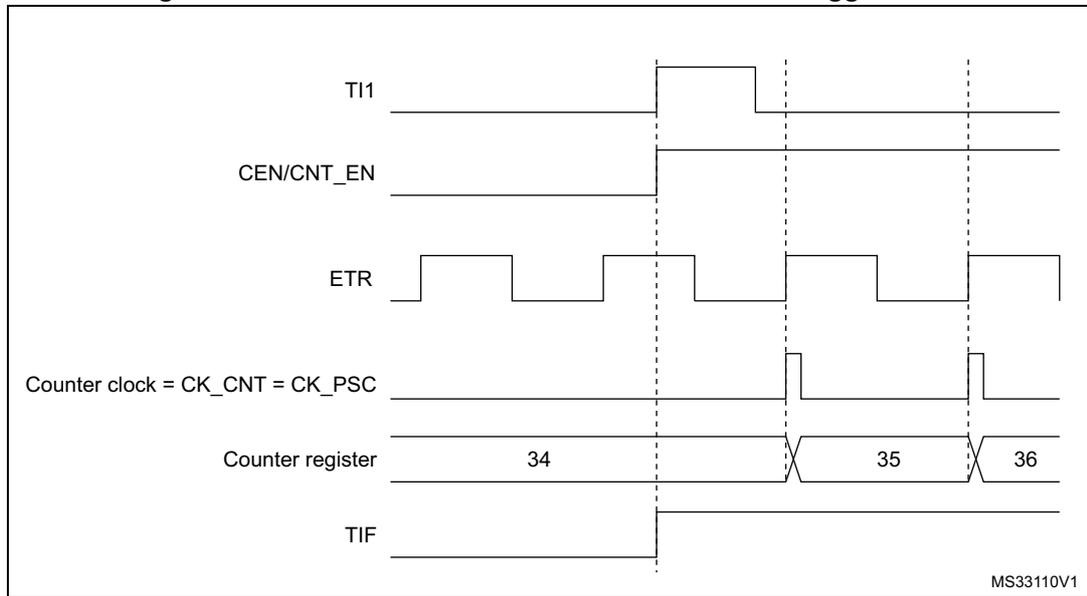
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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 TI1:
  - IC1F=0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S=01 in TIMx\_CCMR1 register to select only the input capture source
  - CC1P=0 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx\_SMCR register.

A rising edge on T11 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 92. Control circuit in external clock mode 2 + trigger mode**



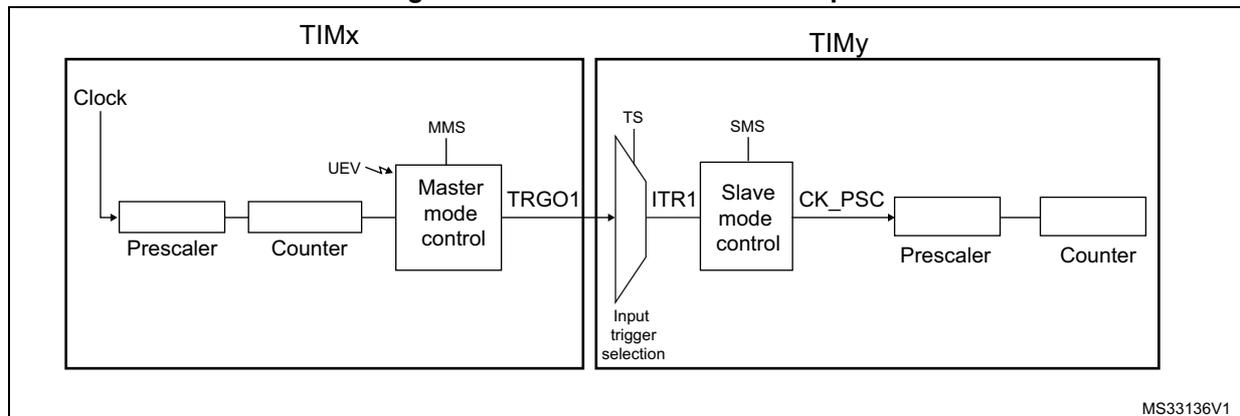
**16.3.15 Timer synchronization**

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 93: 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 93. Master/Slave timer example**



For example, you can configure Timer x to act as a prescaler for Timer y. Refer to [Figure 93](#). To do this:

1. Configure Timer x in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIMx\_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
2. To connect the TRGO1 output of Timer x to Timer y, Timer y must be configured in slave mode using ITR1 as internal trigger. You select this through the TS bits in the TIMy\_SMCR register (writing TS=000).
3. Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIMy\_SMCR register). This causes Timer y to be clocked by the rising edge of the periodic Timer x trigger signal (which correspond to the timer x counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx\_CR1 register).

*Note:* If OCx is selected on Timer x as trigger output (MMS=1xx), its rising edge is used to clock the counter of timer y.

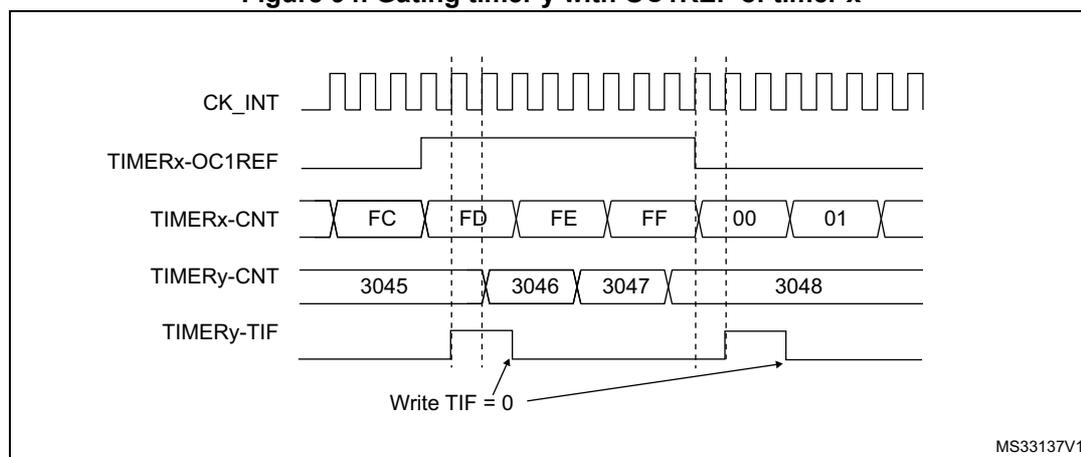
### Using one timer to enable another timer

In this example, we control the enable of Timer y with the output compare 1 of Timer x. Refer to [Figure 93](#) for connections. Timer y counts on the divided internal clock only when OC1REF of Timer x is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

1. Configure Timer x master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIMx\_CR2 register).
2. Configure the Timer x OC1REF waveform (TIMx\_CCMR1 register).
3. Configure Timer y to get the input trigger from Timer x (TS=000 in the TIMy\_SMCR register).
4. Configure Timer y in gated mode (SMS=101 in TIMy\_SMCR register).
5. Enable Timer y by writing '1' in the CEN bit (TIMy\_CR1 register).
6. Start Timer x by writing '1' in the CEN bit (TIMx\_CR1 register).

*Note:* The counter 2 clock is not synchronized with counter 1, this mode only affects the Timer y counter enable signal.

**Figure 94. Gating timer y with OC1REF of timer x**



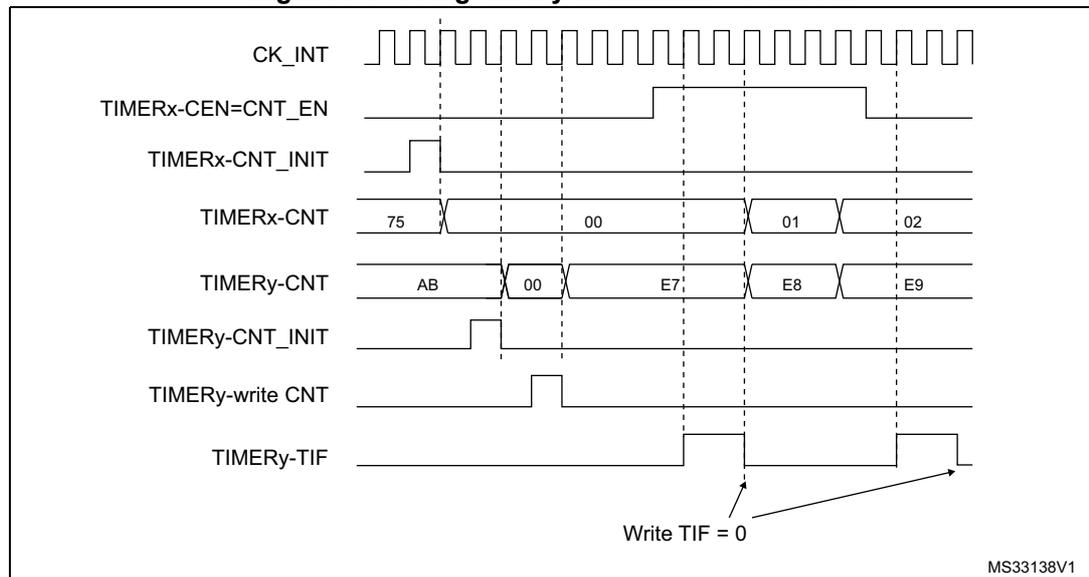
MS33137V1

In the example in [Figure 94](#), the Timer y 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 Timer x. 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 Timer x and Timer y. Timer x is the master and starts from 0. Timer y is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer y stops when Timer x is disabled by writing '0 to the CEN bit in the TIMy\_CR1 register:

1. Configure Timer x master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIMx\_CR2 register).
2. Configure the Timer x OC1REF waveform (TIMx\_CCMR1 register).
3. Configure Timer y to get the input trigger from Timer x (TS=000 in the TIMy\_SMCR register).
4. Configure Timer y in gated mode (SMS=101 in TIMy\_SMCR register).
5. Reset Timer x by writing '1 in UG bit (TIMx\_EGR register).
6. Reset Timer y by writing '1 in UG bit (TIMy\_EGR register).
7. Initialize Timer y to 0xE7 by writing '0xE7' in the timer y counter (TIMy\_CNTL).
8. Enable Timer y by writing '1 in the CEN bit (TIMy\_CR1 register).
9. Start Timer x by writing '1 in the CEN bit (TIMx\_CR1 register).
10. Stop Timer x by writing '0 in the CEN bit (TIMx\_CR1 register).

**Figure 95. Gating timer y with Enable of timer x**

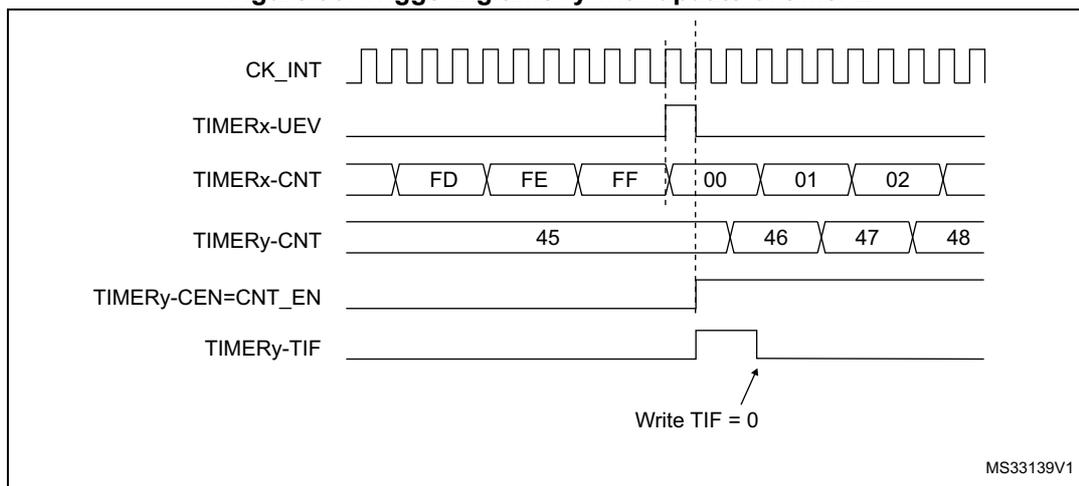


**Using one timer to start another timer**

In this example, we set the enable of Timer y with the update event of Timer x. Refer to [Figure 93](#) for connections. Timer y starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by Timer x. When Timer y receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0 to the CEN bit in the TIM2\_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

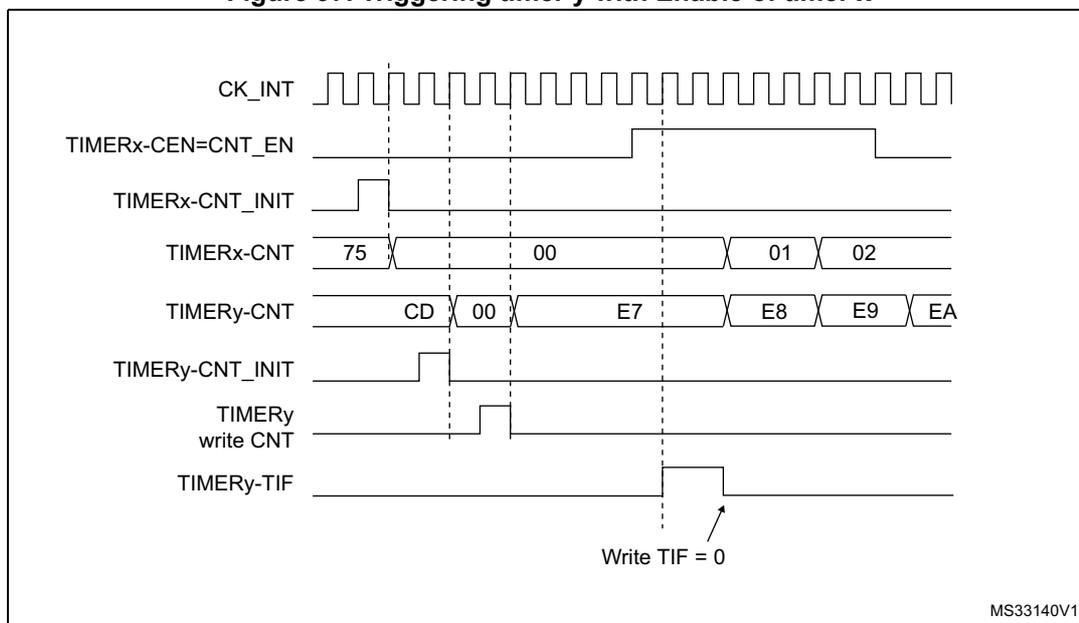
1. Configure Timer x master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIMx\_CR2 register).
2. Configure the Timer x period (TIMx\_ARR registers).
3. Configure Timer y to get the input trigger from Timer x (TS=000 in the TIM2\_SMCR register).
4. Configure Timer y in trigger mode (SMS=110 in TIM2\_SMCR register).
5. Start Timer x by writing '1 in the CEN bit (TIMx\_CR1 register).

**Figure 96. Triggering timer y with update of timer x**



As in the previous example, you can initialize both counters before starting counting. [Figure 97](#) shows the behavior with the same configuration as in [Figure 96](#) but in trigger mode instead of gated mode (SMS=110 in the TIMy\_SMCR register).

**Figure 97. Triggering timer y with Enable of timer x**



### Using one timer as prescaler for another timer

For example, you can configure Timer x to act as a prescaler for Timer y. Refer to [Figure 93](#) for connections. To do this:

1. Configure Timer x master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIMx\_CR2 register). then it outputs a periodic signal on each counter overflow.
2. Configure the Timer x period (TIMx\_ARR registers).
3. Configure Timer y to get the input trigger from Timer x (TS=000 in the TIM2\_SMCR register).
4. Configure Timer y in external clock mode 1 (SMS=111 in TIM2\_SMCR register).
5. Start Timer y by writing '1 in the CEN bit (TIMy\_CR1 register).
6. Start Timer x by writing '1 in the CEN bit (TIMx\_CR1 register).

### Starting 2 timers synchronously in response to an external trigger

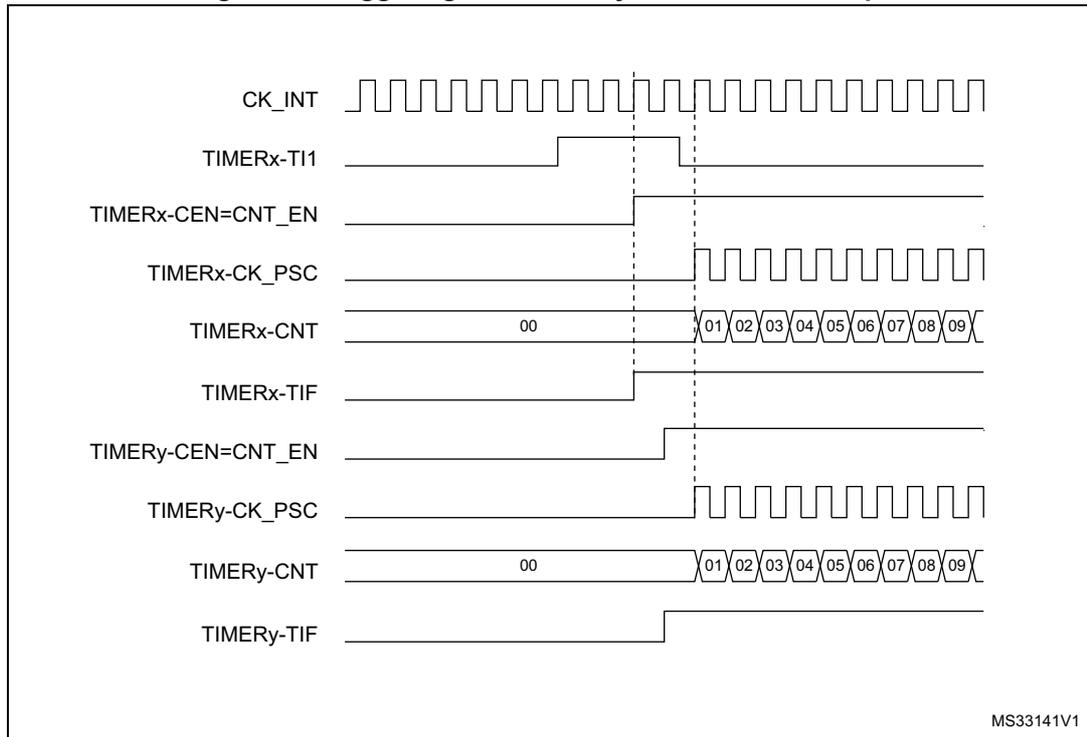
In this example, we set the enable of timer x when its TI1 input rises, and the enable of Timer y with the enable of Timer x. Refer to [Figure 93](#) for connections. To ensure the counters are aligned, Timer x must be configured in Master/Slave mode (slave with respect to TI1, master with respect to Timer y):

1. Configure Timer x master mode to send its Enable as trigger output (MMS=001 in the TIMx\_CR2 register).
2. Configure Timer x slave mode to get the input trigger from TI1 (TS=100 in the TIMx\_SMCR register).
3. Configure Timer x in trigger mode (SMS=110 in the TIMx\_SMCR register).
4. Configure the Timer x in Master/Slave mode by writing MSM=1 (TIMx\_SMCR register).
5. Configure Timer y to get the input trigger from Timer x (TS=000 in the TIM2\_SMCR register).
6. Configure Timer y in trigger mode (SMS=110 in the TIM2\_SMCR register).

When a rising edge occurs on TI1 (Timer x), 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 timer x.

Figure 98. Triggering timer x and y with timer x TI1 input



### 16.3.16 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M4 with FPU 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 31.16.2: Debug support for timers, watchdog, bxCAN and I2C](#).

## 16.4 TIM2 to TIM5/TIM19 registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

The 32-bit peripheral registers have to be written by words (32 bits). All other peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

### 16.4.1 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
Res.	Res.	Res.	Res.	Res.	Res.	CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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, Tlx),

- 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.

### 16.4.2 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
Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.							
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: T11 selection

0: The TIMx\_CH1 pin is connected to T11 input

1: The TIMx\_CH1, CH2 and CH3 pins are connected to the T11 input (XOR combination)

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, must be kept at reset value.

### 16.4.3 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]			Res.	SMS[2:0]		
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or  $\overline{\text{ETR}}$  is used for trigger operations

- 0: ETR is noninverted, 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:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=2
- 0010:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=4
- 0011:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , N=8
- 0100:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$ , N=6
- 0101:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/2$ , N=8
- 0110:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$ , N=6
- 0111:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/4$ , N=8
- 1000:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$ , N=6
- 1001:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/8$ , N=8
- 1010:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=5
- 1011:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=6
- 1100:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$ , N=8
- 1101:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=5
- 1110:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , N=6
- 1111:  $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$ , 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: T11 Edge Detector (TI1F\_ED)

101: Filtered Timer Input 1 (TI1FP1)

110: Filtered Timer Input 2 (TI2FP2)

111: External Trigger input (ETRF)

See [Table 45: TIMx internal trigger connection on page 340](#) 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 '1'.

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 45. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM19	TIM15	TIM3	TIM14
TIM3	TIM19	TIM2	TIM5	TIM14
TIM4	TIM19	TIM2	TIM3	TIM15
TIM5	TIM2	TIM3	TIM4	TIM15
TIM19	TIM2	TIM3	TIM15	TIM16

16.4.4 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, must be kept at reset value.
- Bit 14 **TDE**: Trigger DMA request enable  
 0: Trigger DMA request disabled.  
 1: Trigger DMA request enabled.
- Bit 13 Reserved, 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, must be kept at reset value.
- Bit 6 **TIE**: Trigger interrupt enable  
 0: Trigger interrupt disabled.  
 1: Trigger interrupt enabled.
- Bit 5 Reserved, must be kept at reset value.



- Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable  
 0: CC4 interrupt disabled.  
 1: CC4 interrupt enabled.
- Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable  
 0: CC3 interrupt disabled  
 1: CC3 interrupt enabled
- Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable  
 0: CC2 interrupt disabled  
 1: CC2 interrupt enabled
- Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable  
 0: CC1 interrupt disabled  
 1: CC1 interrupt enabled
- Bit 0 **UIE**: Update interrupt enable  
 0: Update interrupt disabled  
 1: Update interrupt enabled

### 16.4.5 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
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	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, must be kept at reset value.
- Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag  
 refer to CC1OF description
- Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag  
 refer to CC1OF description
- Bit 10 **CC2OF**: Capture/compare 2 overcapture flag  
 refer to CC1OF description
- Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
 This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
 0: No overcapture has been detected  
 1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set
- Bits 8:7 Reserved, must be kept at reset value.
- Bit 6 **TIF**: Trigger interrupt flag  
 This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
 0: No trigger event occurred  
 1: Trigger interrupt pending
- Bit 5 Reserved, must be kept at reset value.

- Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag  
refer to CC1IF description
- Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag  
refer to CC1IF description
- Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag  
refer to CC1IF description
- Bit 1 **CC1IF**: Capture/compare 1 interrupt flag  
**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 TIM5 and TIM19) 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.

### 16.4.6 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
Res.	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).

### 16.4.7 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. 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 15:12 **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  $f_{DTS}$  1000:  $f_{SAMPLING}=f_{DTS}/8$ , N=6

0001:  $f_{SAMPLING}=f_{CK\_INT}$ , N=2 1001:  $f_{SAMPLING}=f_{DTS}/8$ , N=8

0010:  $f_{SAMPLING}=f_{CK\_INT}$ , N=4 1010:  $f_{SAMPLING}=f_{DTS}/16$ , N=5

0011:  $f_{SAMPLING}=f_{CK\_INT}$ , N=8 1011:  $f_{SAMPLING}=f_{DTS}/16$ , N=6

0100:  $f_{SAMPLING}=f_{DTS}/2$ , N=6 1100:  $f_{SAMPLING}=f_{DTS}/16$ , N=8

0101:  $f_{SAMPLING}=f_{DTS}/2$ , N=8 1101:  $f_{SAMPLING}=f_{DTS}/32$ , N=5

0110:  $f_{SAMPLING}=f_{DTS}/4$ , N=6 1110:  $f_{SAMPLING}=f_{DTS}/32$ , N=6

0111:  $f_{SAMPLING}=f_{DTS}/4$ , N=8 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).*

### 16.4.8 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).*

### 16.4.9 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
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw												

Bit 15 **CC4NP**: Capture/Compare 4 output Polarity.

Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output Polarity.

refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable.

refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 output Polarity.

refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/Compare 3 output Polarity.

refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable.

refer to CC1E description

- Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*  
refer to CC1NP description
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*  
refer to CC1P description
- Bit 4 **CC2E**: *Capture/Compare 2 output enable.*  
refer to CC1E description
- Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*  
  - CC1 channel configured as output:**  
CC1NP must be kept cleared in this case.
  - CC1 channel configured as input:**  
This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.
- Bit 2 Reserved, must be kept at reset value.
- Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*  
  - 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.  
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 46. 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.

### 16.4.10 TIMx counter (TIMx\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16] (depending on timers)															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **CNT[31:16]**: High counter value (on TIM2 and TIM5).

Bits 15:0 **CNT[15:0]**: Low counter value

### 16.4.11 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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.

### 16.4.12 TIMx auto-reload register (TIMx\_ARR)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16] (depending on timers)															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **ARR[31:16]**: High auto-reload value (on TIM2 and TIM5).

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 16.3.1: Time-base unit on page 296](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 16.4.13 TIMx capture/compare register 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16] (depending on timers)															
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
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (on TIM2 and TIM5).

Bits 15:0 **CCR1[15:0]**: Low 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).

### 16.4.14 TIMx capture/compare register 2 (TIMx\_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16] (depending on timers)															
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
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (on TIM2 and TIM5).

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

**If channel CC2 is configured as output:**

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_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 signaled on OC2 output.

**If channel CC2 is configured as input:**

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

### 16.4.15 TIMx capture/compare register 3 (TIMx\_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16] (depending on timers)															
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
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (on TIM2 and TIM5).

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

**If channel CC3 is configured as output:**

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_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).

### 16.4.16 TIMx capture/compare register 4 (TIMx\_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16] (depending on timers)															
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
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (on TIM2 and TIM5).

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_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).

### 16.4.17 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
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

- Example:
- 00000: TIMx\_CR1,
  - 00001: TIMx\_CR2,
  - 00010: TIMx\_SMCR,
  - ...

**Example:** Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx\_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx\_CR1 address.

### 16.4.18 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  $(\text{TIMx\_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

#### 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.

### 16.4.19 TIM2 option register (TIM2\_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ITR1_RMP		Res.									
				rw	rw										

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

Bits 11:10 **ITR1\_RMP**: Internal trigger 1 remap

Set and cleared by software.

00: TIM8\_TRGOUT

01: PTP trigger output is connected to TIM2\_ITR1

10: OTG FS SOF is connected to the TIM2\_ITR1 input

11: OTG HS SOF is connected to the TIM2\_ITR1 input

Bits 9:0 Reserved, must be kept at reset value.

### 16.4.20 TIMx register map

TIMx registers are mapped as described in the table below:

**Table 47. TIM2 to TIM15/19 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	<b>TIMx_CR1</b>	Res.	Res.	Res.	Res.	CKD [1:0]	Res.	ARPE	CMS [1:0]	Res.	DIR	OPM	URS	UDIS	CEN																			
	Reset value																								0	0	0	0	0	0	0	0	0	0
0x04	<b>TIMx_CR2</b>	Res.	Res.	Res.	Res.	Res.	Res.	T1S	MMS[2:0]	Res.	CCDS	Res.	Res.	Res.																				
	Reset value																									0	0	0	0	0	0	0	0	0
0x08	<b>TIMx_SMCR</b>	Res.	ETP	ECE	ETPS [1:0]			ETF [3:0]			MSM	TS[2:0]		Res.	SMS[2:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	<b>TIMx_DIER</b>	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE																	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	<b>TIMx_SR</b>	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF																		
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	<b>TIMx_EGR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG																			
	Reset value																										0	0	0	0	0	0	0	0

Table 47. TIM2 to TIM15/19 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x18	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x24	TIMx_CNT	CNT[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CNT[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	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[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	0
0x30	Res.																																	
0x34	TIMx_CCR1	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[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	0
0x38	TIMx_CCR2	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[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	0
0x3C	TIMx_CCR3	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[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	0
0x40	TIMx_CCR4	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[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	0
0x44	Res.																																	



Table 47. TIM2 to TIM15/19 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
0x48	TIMx_DCR	Res.	Res.	Res.	Res.	DBL[4:0]				Res.	Res.	Res.	DBA[4:0]																				
	Reset value																					0	0	0	0	0				0	0	0	0
0x4C	TIMx_DMAR	Res.	DMAB[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TIM2_OR	Res.	Res.	Res.	Res.	ITR1 _R _P		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																	
	Reset value																						0	0									

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 17 General-purpose timers (TIM12/13/14)

### 17.1 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 17.4.12](#).

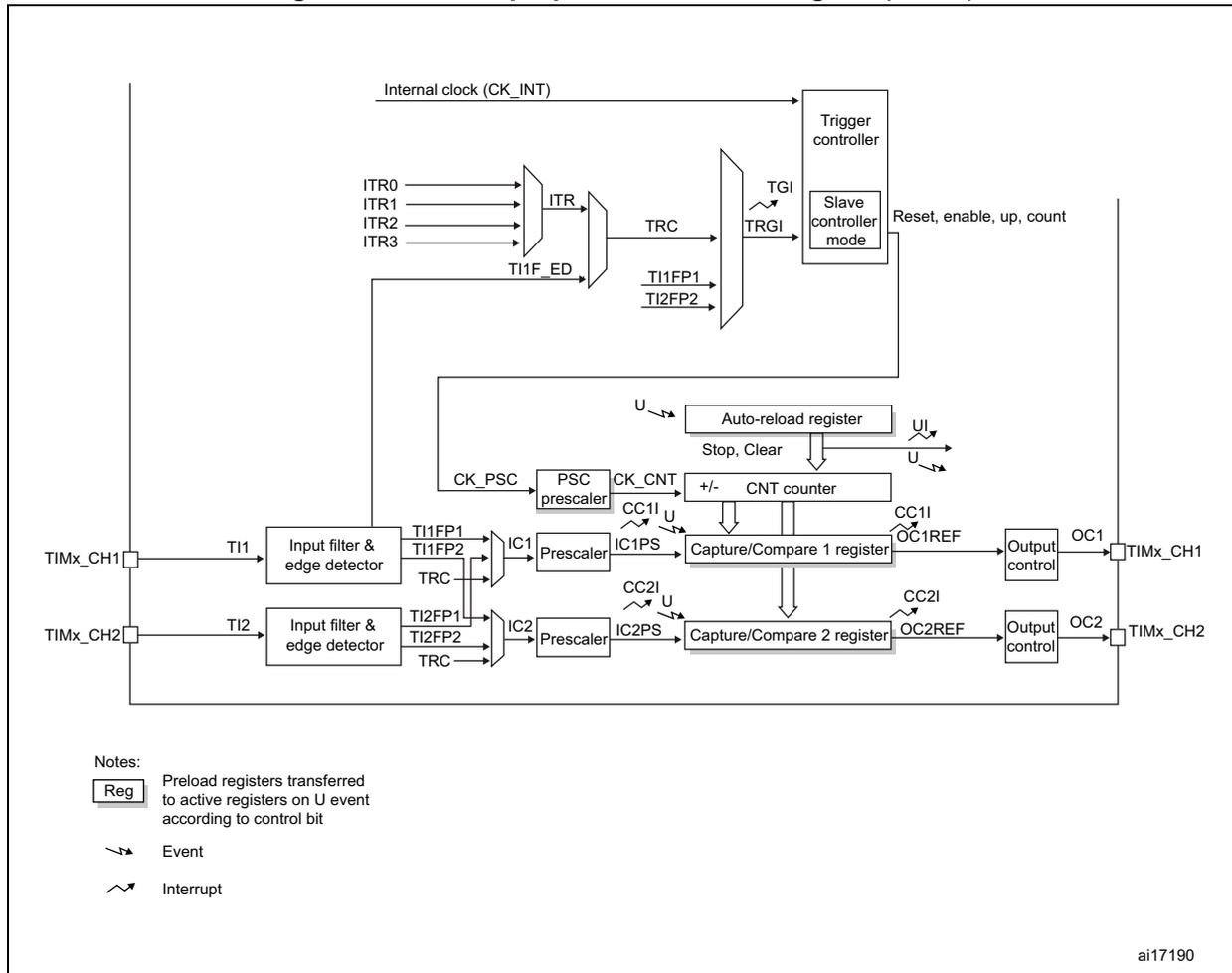
### 17.2 TIM12/13/14 main features

#### 17.2.1 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit auto-reload upcounter (in medium density devices)
- 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 99. General-purpose timer block diagram (TIM12)

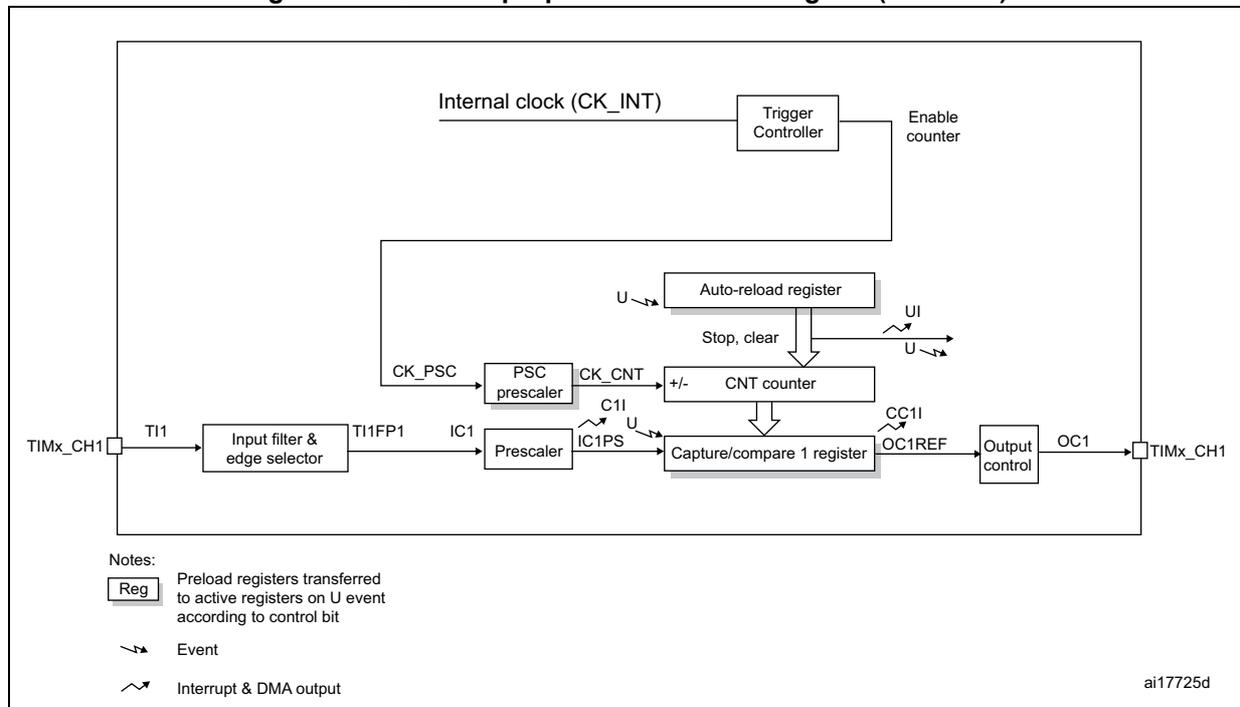


### 17.3 TIM13/TIM14 main features

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 100. General-purpose timer block diagram (TIM13/14)**



## 17.4 TIM12/13/14 functional description

### 17.4.1 Time-base unit

The main block of the timer is a 16-bit counter with its related auto-reload register. The counter counts 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.

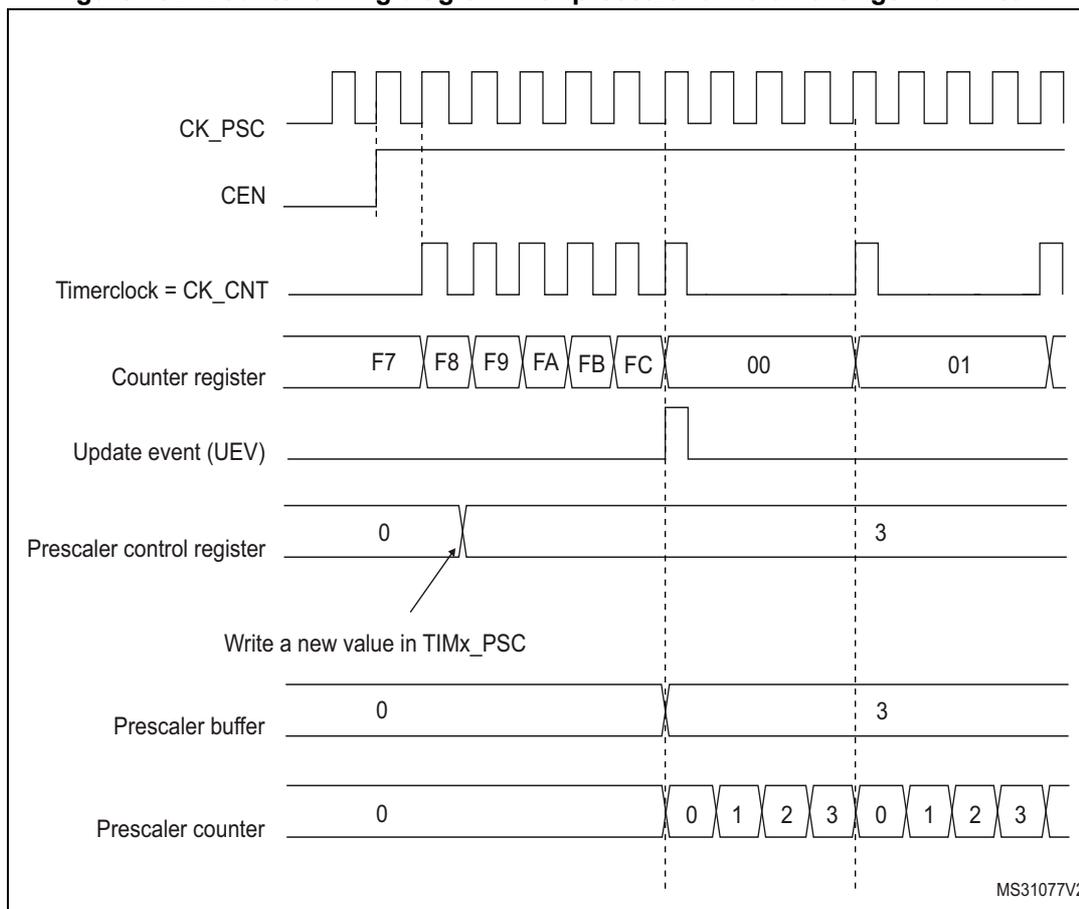
#### Prescaler description

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 102* and *Figure 103* give some examples of the counter behavior when the prescaler ratio is changed on the fly.



Figure 102. Counter timing diagram with prescaler division change from 1 to 4



## 17.4.2 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.

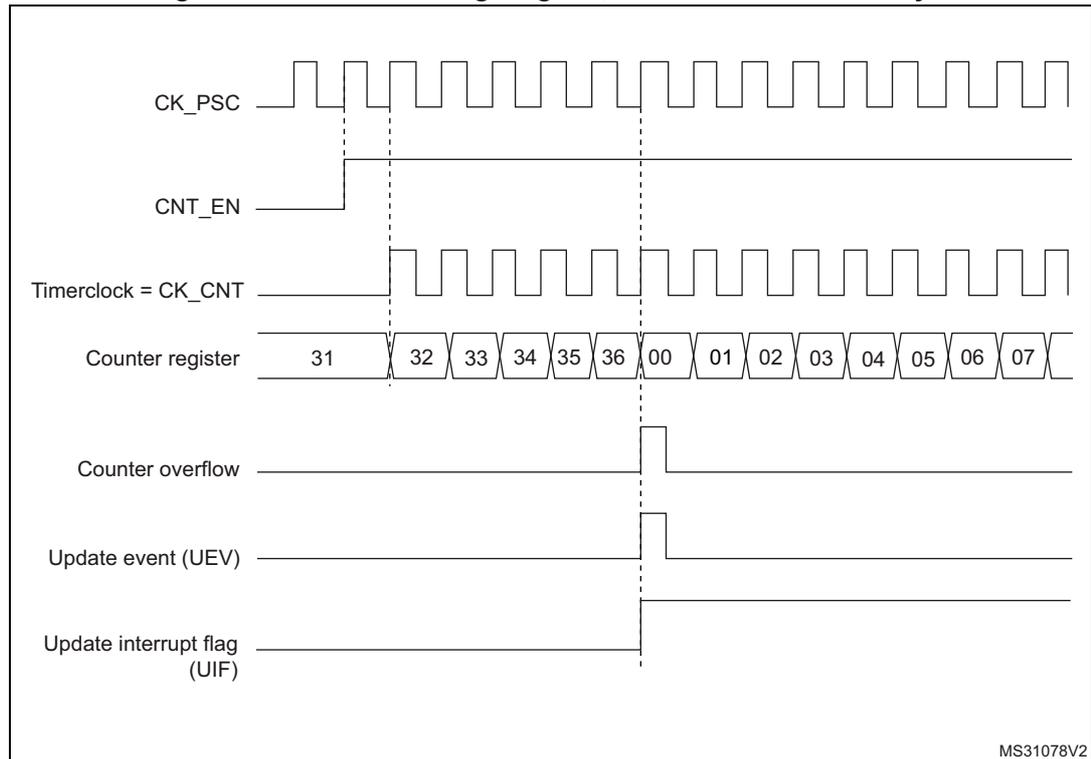
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as 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):

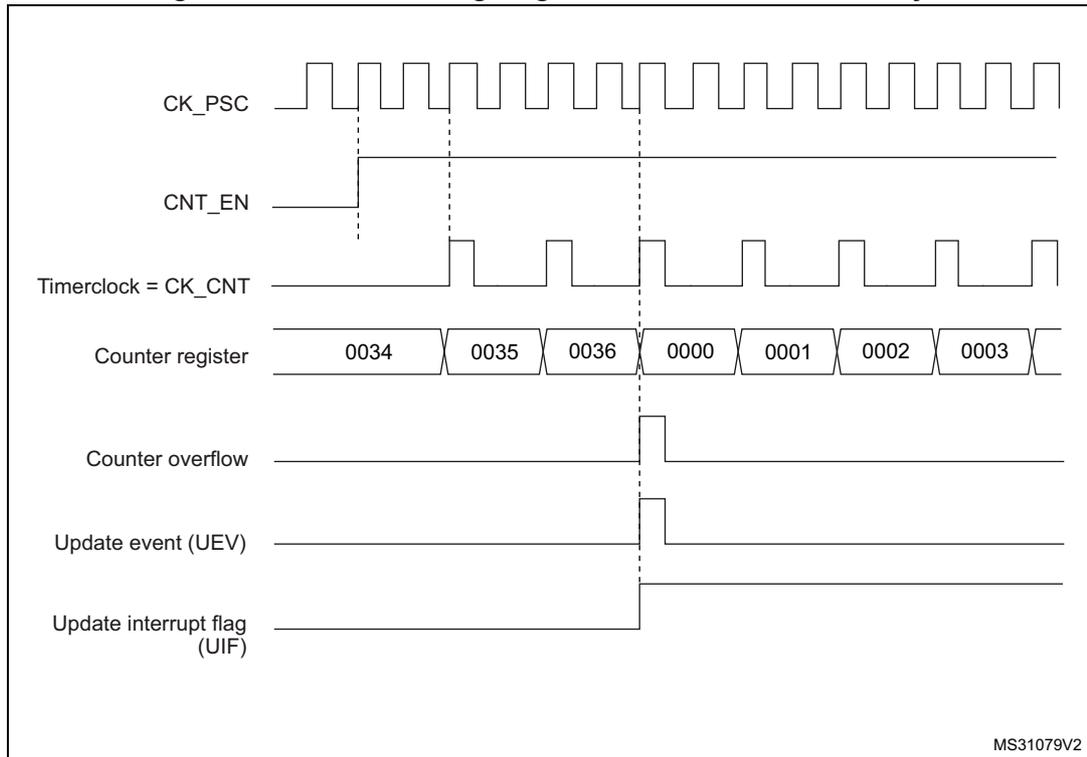
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 103. Counter timing diagram, internal clock divided by 1**



**Figure 104. Counter timing diagram, internal clock divided by 2**



**Figure 105. Counter timing diagram, internal clock divided by 4**

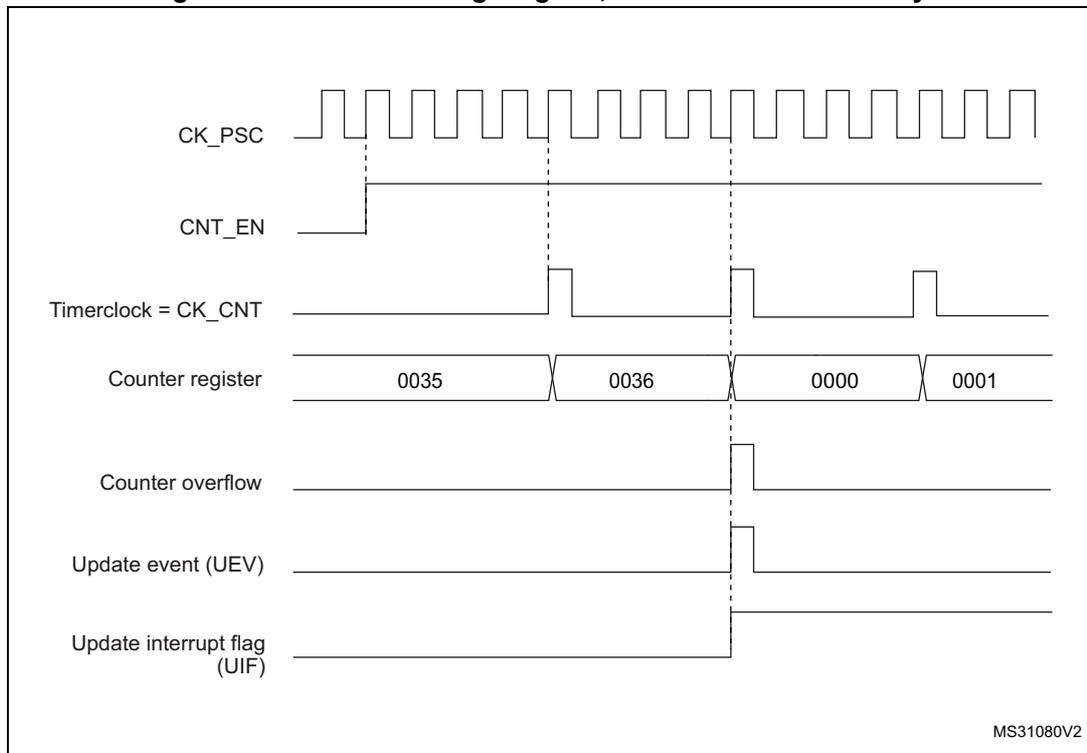


Figure 106. Counter timing diagram, internal clock divided by N

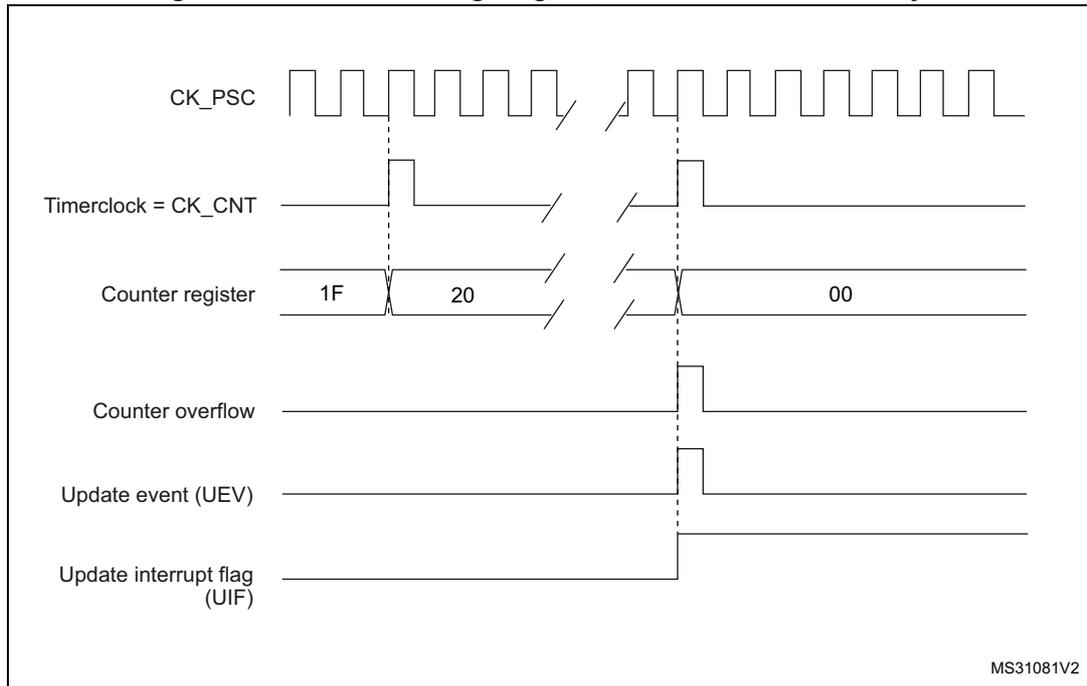
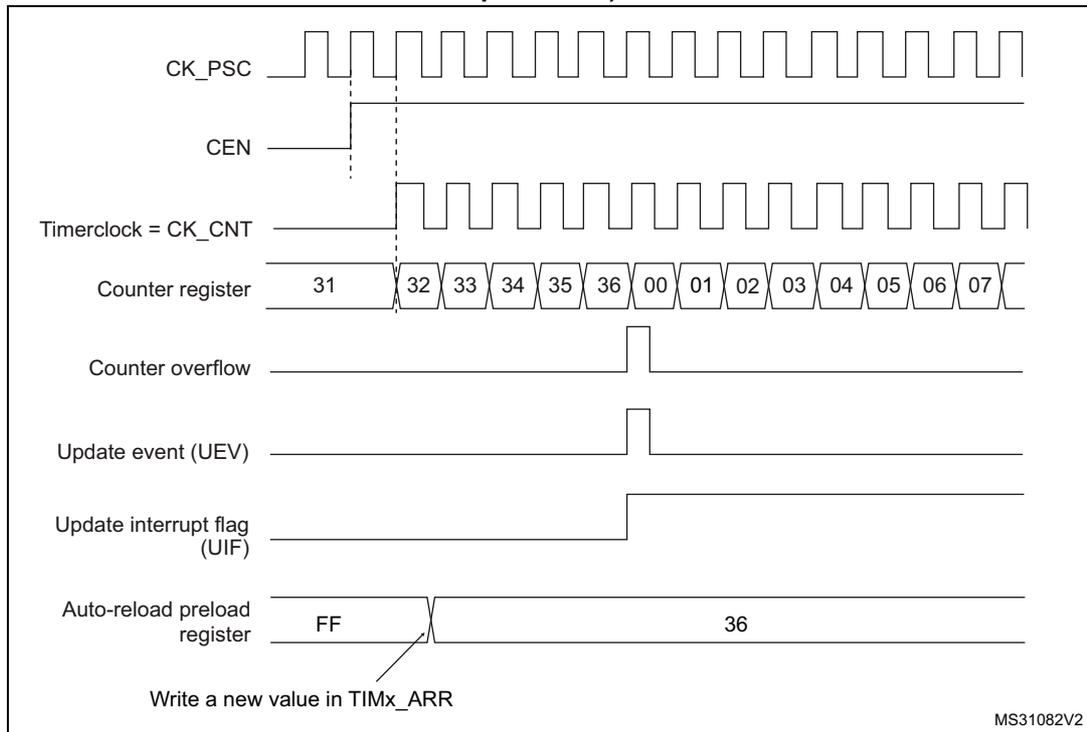
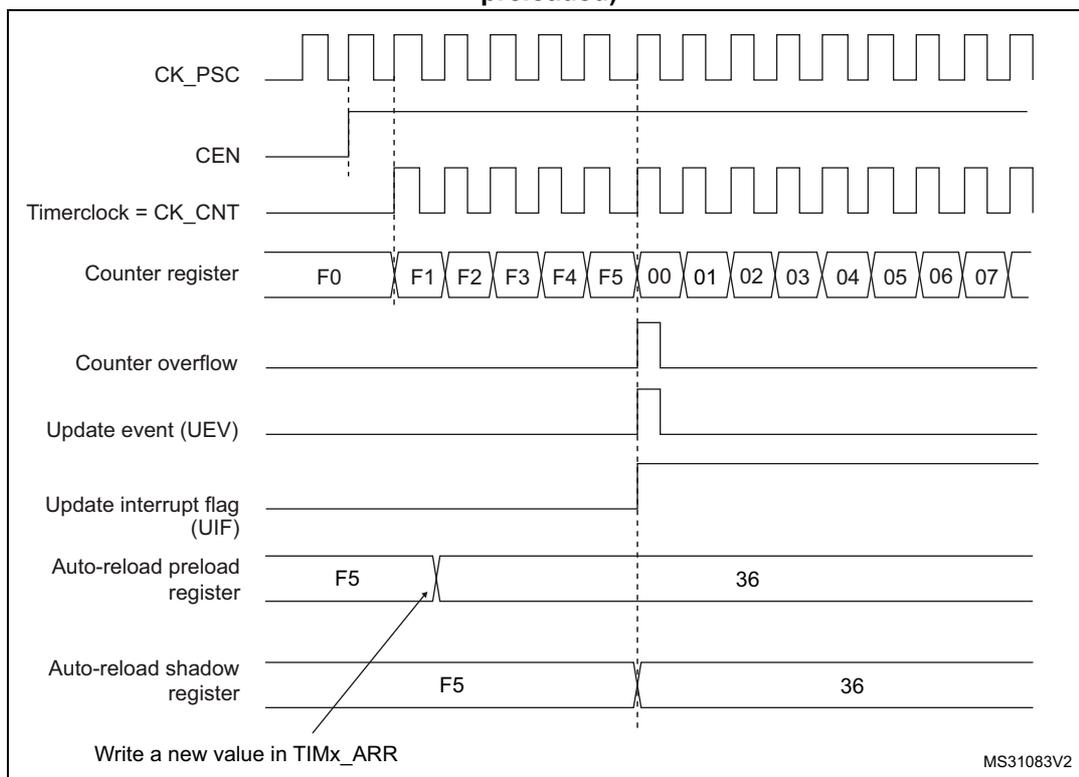


Figure 107. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)



**Figure 108. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 17.4.3 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 [Section : Using one timer as prescaler for another](#) for more details.

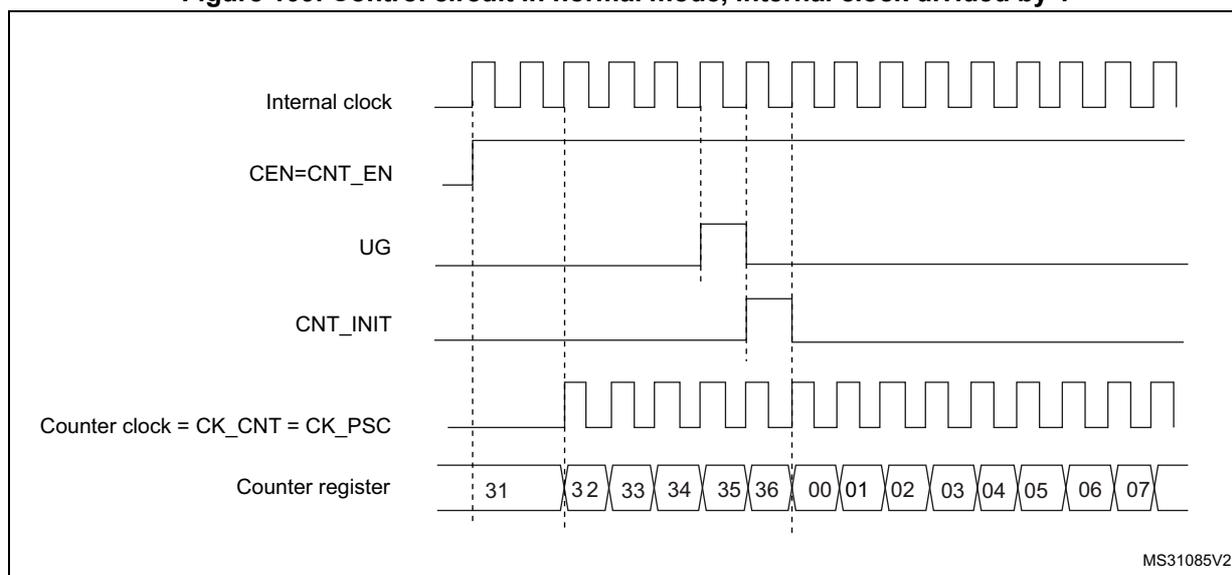
#### Internal clock source (CK\_INT)

The internal clock source is the default clock source for TIM13/TIM14.

For 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 109](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

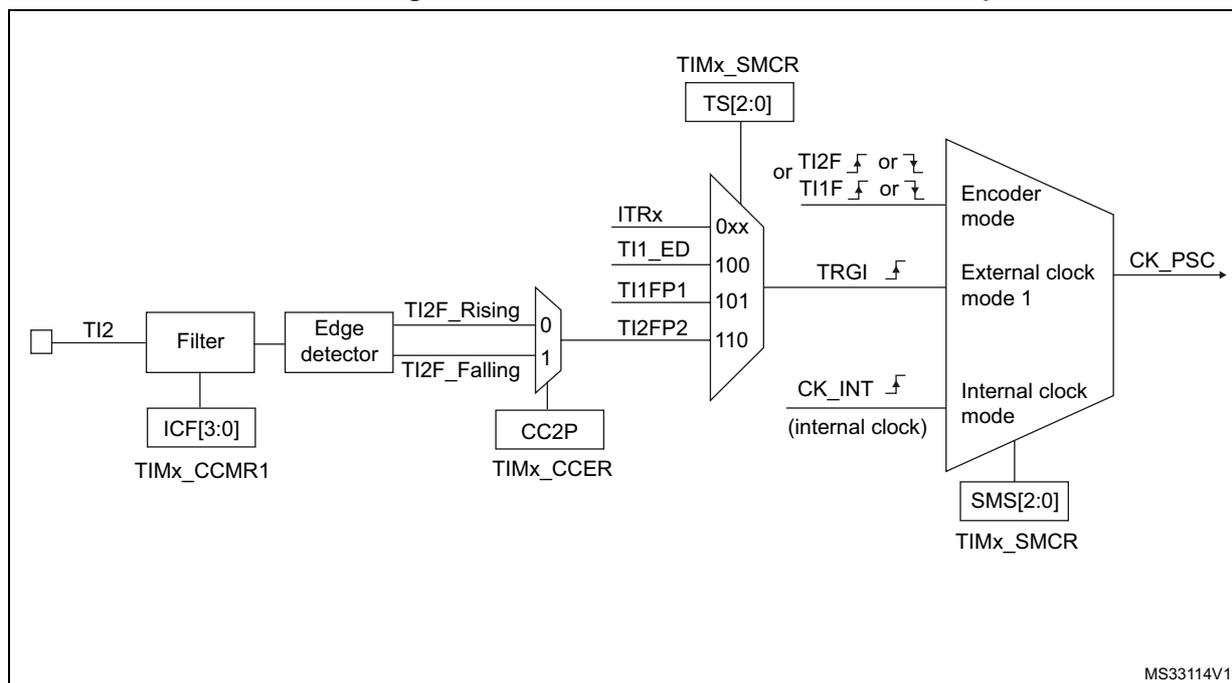
Figure 109. Control circuit in normal mode, internal clock divided by 1



**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 110. 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:

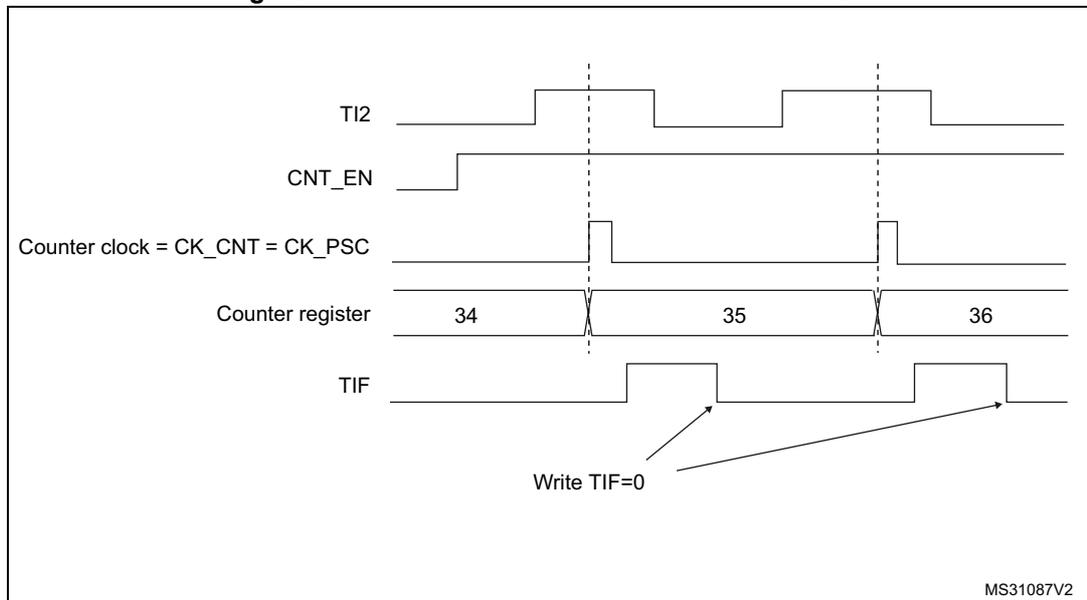
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F='0000').
3. Select the rising edge polarity by writing 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 111. Control circuit in external clock mode 1**



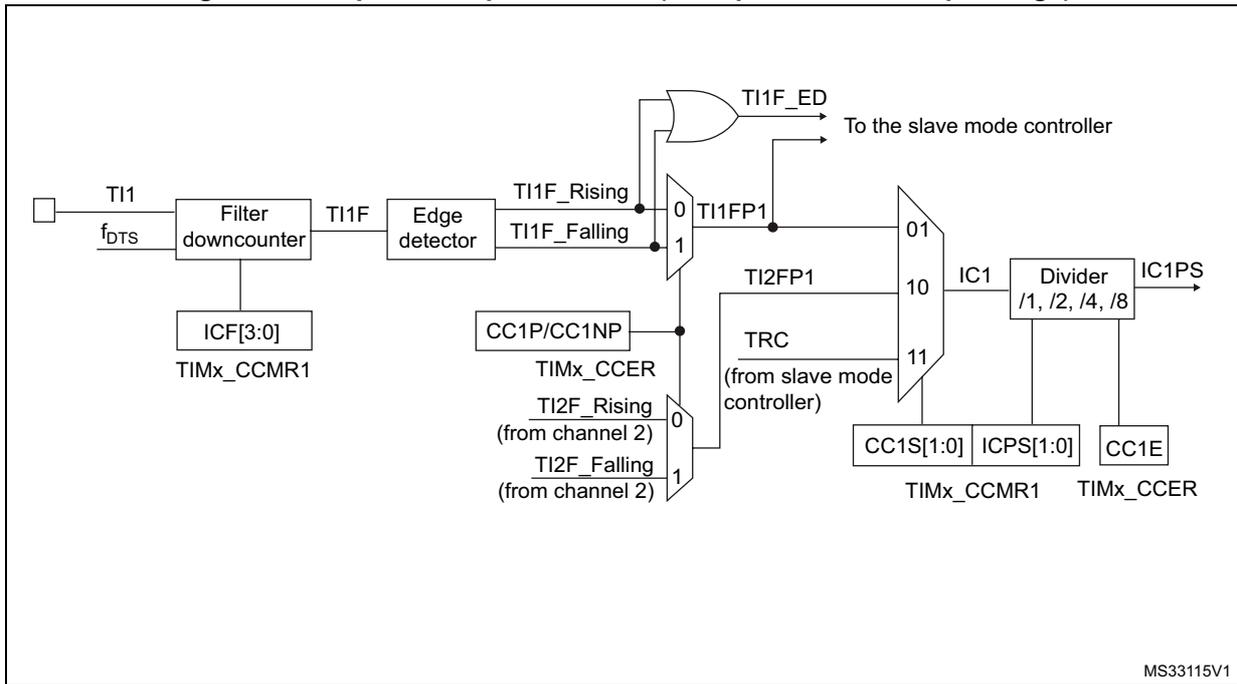
#### 17.4.4 Capture/compare channels

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 112 to Figure 114* 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 112. 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 113. Capture/compare channel 1 main circuit

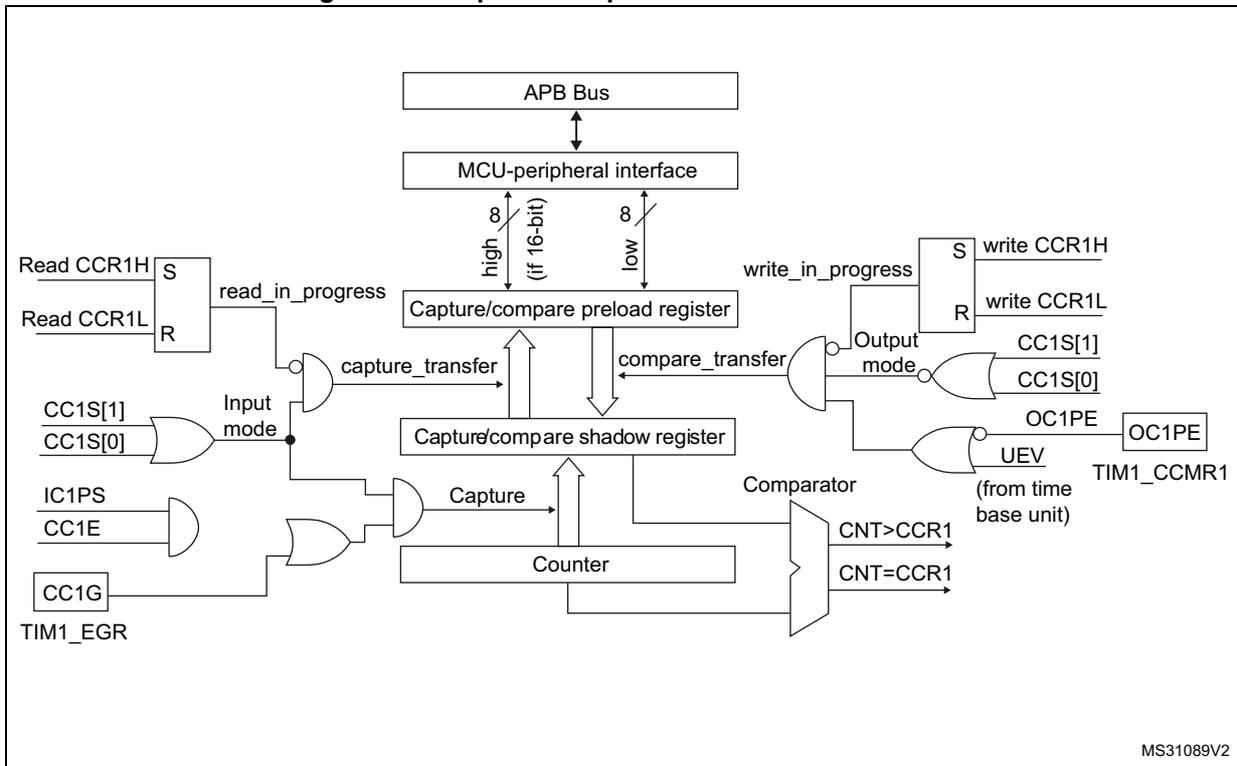
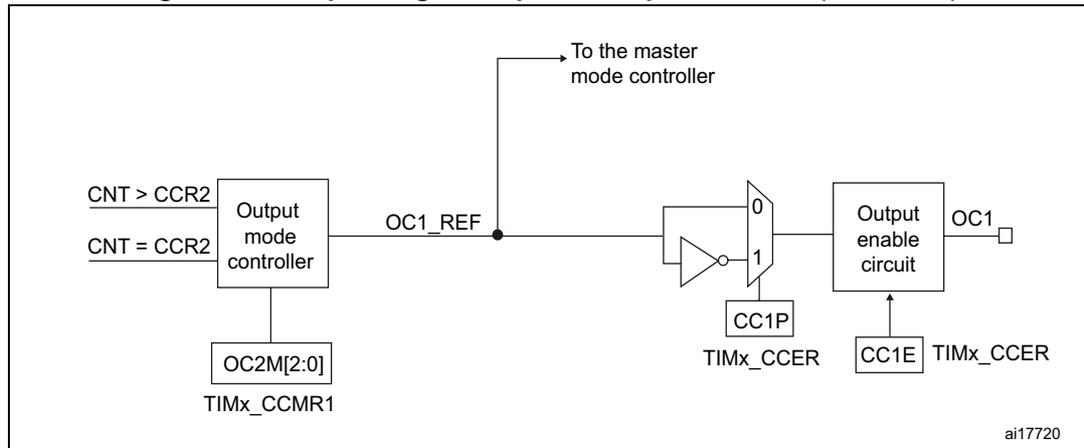


Figure 114. Output stage of capture/compare channel (channel 1)



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.

### 17.4.5 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 most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to '0011' in the TIMx\_CCMR1 register.

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.

#### 17.4.6 PWM input mode (only for TIM12)

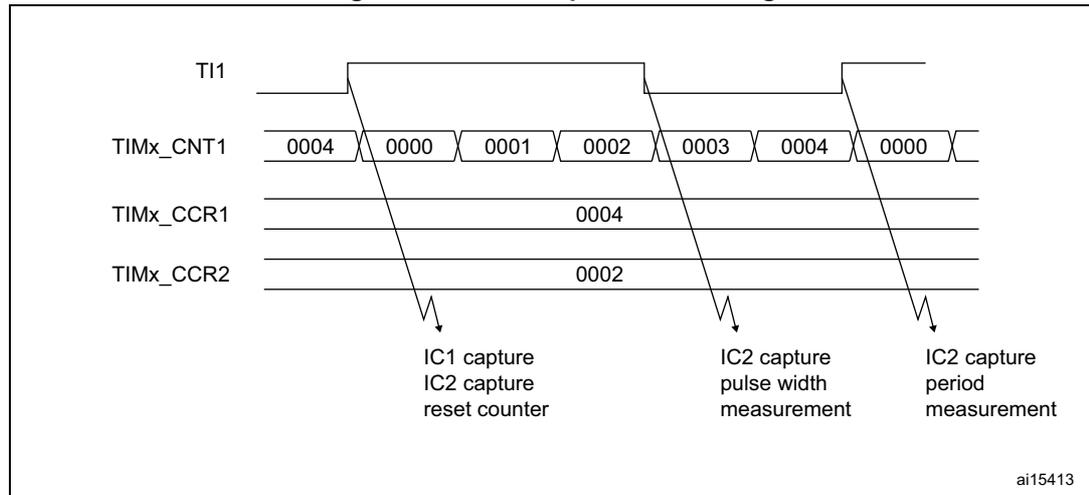
This mode is a particular case of input capture mode. The procedure is the same except:

- 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 115. 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.

### 17.4.7 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.

### 17.4.8 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.

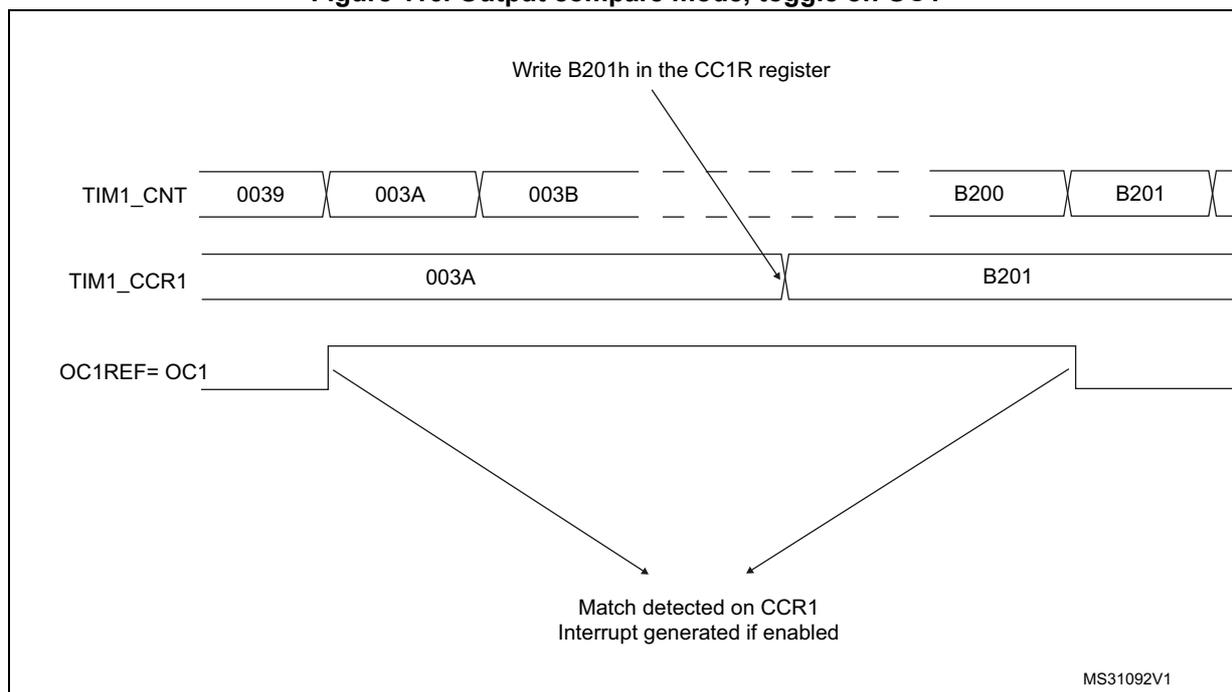
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 116](#).

**Figure 116. Output compare mode, toggle on OC1**



### 17.4.9 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.

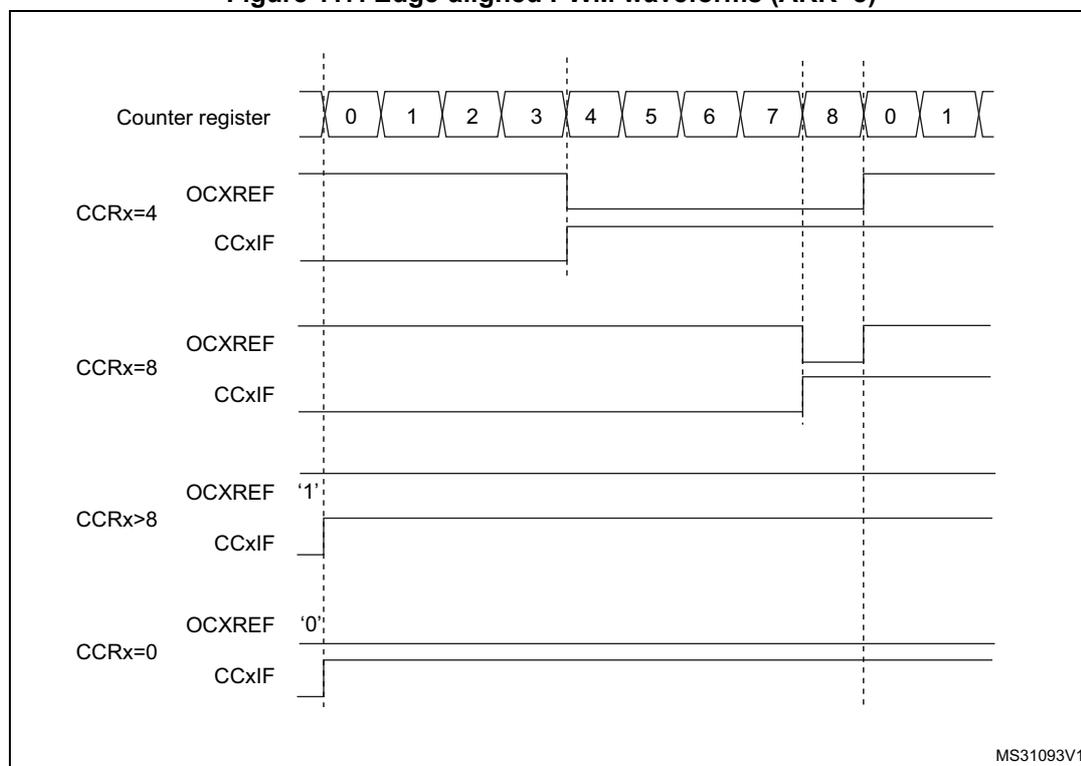
In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $TIMx\_CNT \leq TIMx\_CCRx$ .

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 117* shows some edge-aligned PWM waveforms in an example where  $TIMx\_ARR=8$ .

**Figure 117. Edge-aligned PWM waveforms (ARR=8)**



MS31093V1

### 17.4.10 One-pulse mode (only for TIM12)

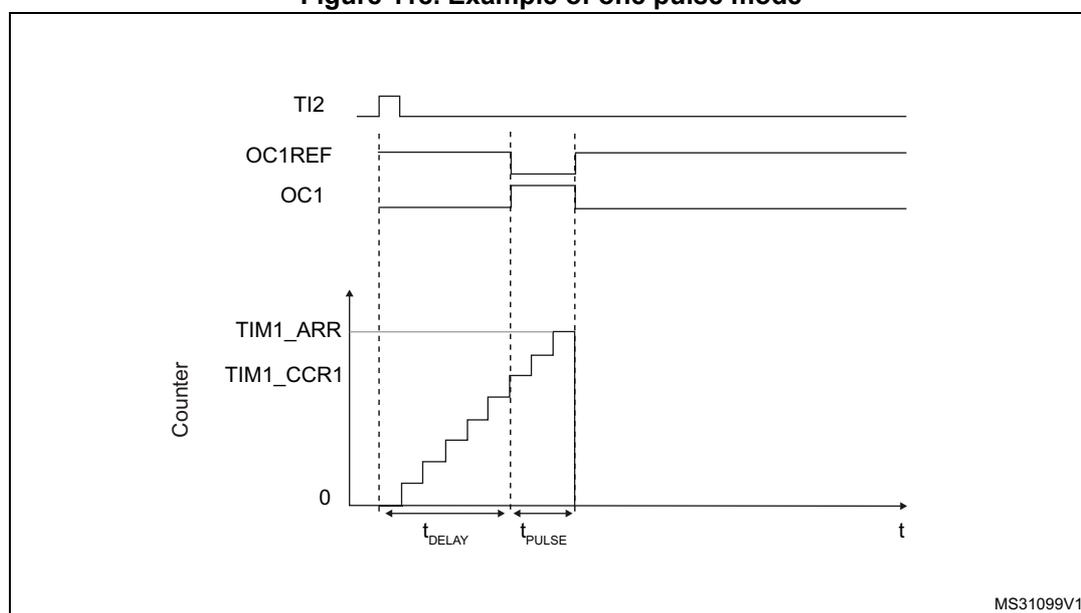
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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:

$$CNT < CCRx \leq ARR \text{ (in particular, } 0 < CCRx)$$

Figure 118. 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  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say 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 Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY min}}$  we can get.

If 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.

#### 17.4.11 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.

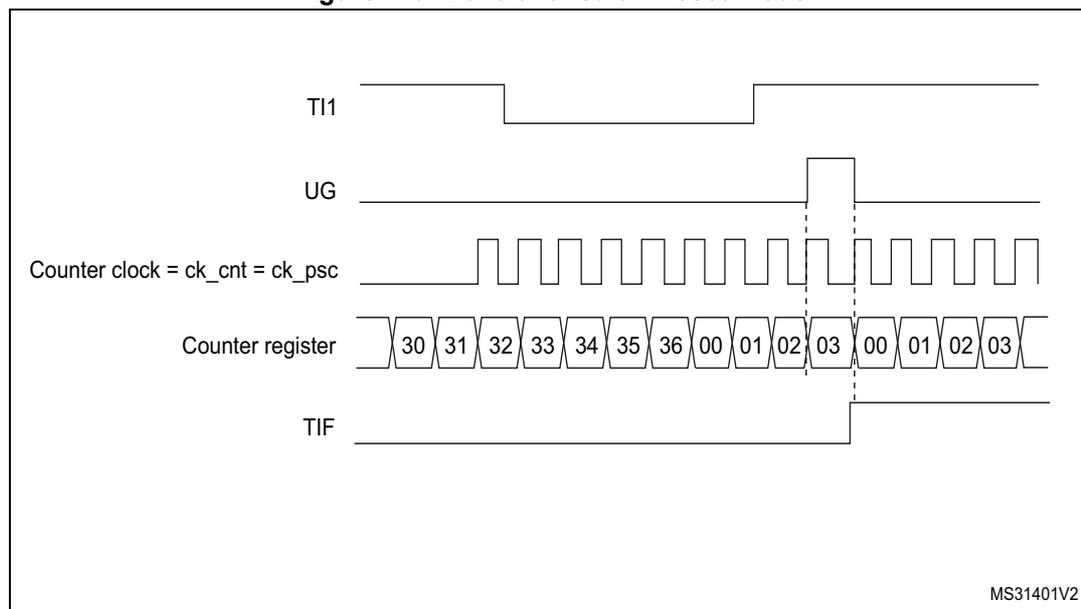
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

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 119. Control circuit in reset mode



MS31401V2

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

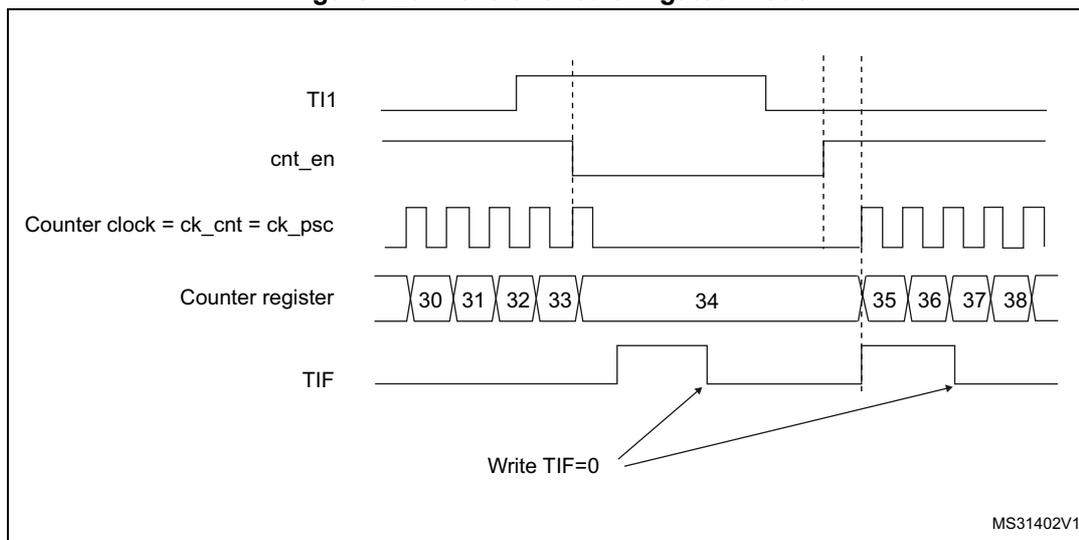
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we 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 120. Control circuit in gated mode



**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

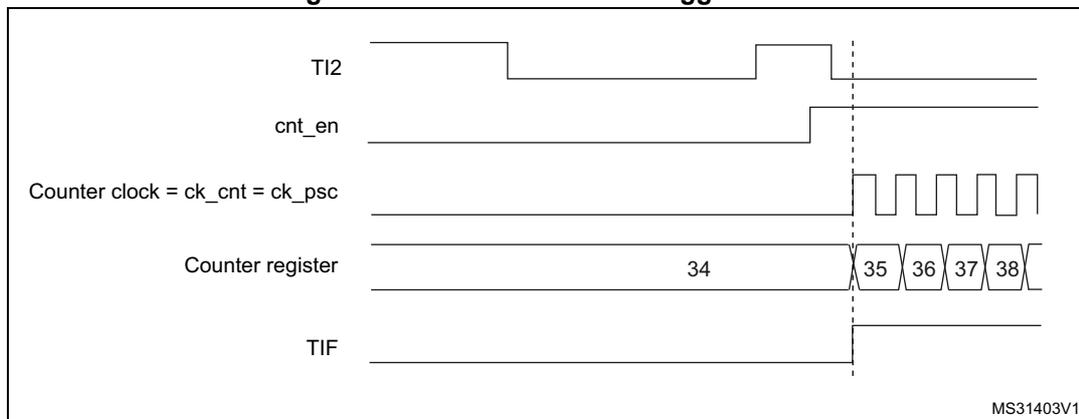
In the following example, the upcounter starts in response to a rising edge on TI2 input:

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 121. Control circuit in trigger mode



### 17.4.12 Timer synchronization (TIM12)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to [Section 16.3.15: Timer synchronization on page 329](#) for details.

### 17.4.13 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M4 with FPU 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 31.16.2: Debug support for timers, watchdog, bxCAN and I2C](#).

## 17.5 TIM12 registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

### 17.5.1 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	
Res.						CKD[1:0]		ARPE	Res.				OPM	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 (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, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

- 0: Counter is not stopped on the update event
- 1: Counter stops counting on the next update event (clearing the CEN bit).

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.

### 17.5.2 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	
Res.								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 53: TIMx Internal trigger connection on page 441](#) 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 48. TIMx Internal trigger connection<sup>(1)</sup>**

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.

### 17.5.3 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
Res.									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.

### 17.5.4 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
Res.				CC2OF	CC1OF	Res.			TIF	Res.			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, must be kept at reset value.

- Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag  
refer to CC1IF description
- Bit 1 **CC1IF**: Capture/compare 1 interrupt flag  
**If channel CC1 is configured as output:**  
This flag is set by hardware when the counter matches the compare value. It is cleared by software.  
0: No match.  
1: The content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the contents of TIMx\_CCR1 are greater than the contents of TIMx\_ARR, the CC1IF bit goes high on the counter overflow.  
**If channel CC1 is configured as input:**  
This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx\_CCR1 register.  
0: No input capture occurred.  
1: The counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).
- Bit 0 **UIF**: Update interrupt flag  
This bit is set by hardware on an update event. It is cleared by software.  
0: No update occurred.  
1: Update interrupt pending. This bit is set by hardware when the registers are updated:  
– At overflow 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.

### 17.5.5 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
Res.									TG	Res.			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.

### 17.5.6 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, must be kept at reset value.

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, must be kept at reset value.

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  $f_{DTS}$  1000:  $f_{SAMPLING}=f_{DTS}/8$ , N=6

0001:  $f_{SAMPLING}=f_{CK\_INT}$ , N=2 1001:  $f_{SAMPLING}=f_{DTS}/8$ , N=8

0010:  $f_{SAMPLING}=f_{CK\_INT}$ , N=4 1010:  $f_{SAMPLING}=f_{DTS}/16$ , N=5

0011:  $f_{SAMPLING}=f_{CK\_INT}$ , N=8 1011:  $f_{SAMPLING}=f_{DTS}/16$ , N=6

0100:  $f_{SAMPLING}=f_{DTS}/2$ , N=6 1100:  $f_{SAMPLING}=f_{DTS}/16$ , N=8

0101:  $f_{SAMPLING}=f_{DTS}/2$ , N=8 1101:  $f_{SAMPLING}=f_{DTS}/32$ , N=5

0110:  $f_{SAMPLING}=f_{DTS}/4$ , N=6 1110:  $f_{SAMPLING}=f_{DTS}/32$ , N=6

0111:  $f_{SAMPLING}=f_{DTS}/4$ , N=8 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  $ICx[3:0]=1, 2$  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).*

### 17.5.7 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
Res.								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								r/w		r/w	r/w	r/w		r/w	r/w

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 49. 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.

**17.5.8 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

**17.5.9 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.

**17.5.10 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 17.4.1: Time-base unit on page 361](#) for more details about ARR update and behavior.  
 The counter is blocked while the auto-reload value is null.

### 17.5.11 TIM12 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 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 CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

### 17.5.12 TIM12 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 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).

17.5.13 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 50. 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	Res.	CKD [1:0]	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN																								
	Reset value																							0	0	0				0	0	0	0		
0x04	TIMx_CR2	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.																									
	Reset value																											0	0	0					
0x08	TIMx_SMCR	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]																										
	Reset value																										0	0	0				0	0	0
0x0C	TIMx_DIER	Res.	Res.	Res.	TIE	Res.	Res.	Res.	Res.	CC2IE	CC1IE	UIE																							
	Reset value																											0				0	0	0	
0x10	TIMx_SR	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	Res.	CC2IF	CC1IF	UIF																					
	Reset value																							0	0			0				0	0	0	
0x14	TIMx_EGR	Res.	Res.	Res.	TG	Res.	Res.	Res.	Res.	CC2G	CC1G	UG																							
	Reset value																											0				0	0	0	
0x18	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																		
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
Reset value																																			
0x1C	Res.																																		
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																		
0x24	TIMx_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																		
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																		



Table 50. 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
0x2C	TIMx_ARR	Res.	ARR[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Res.																																
0x34	TIMx_CCR1	Res.	CCR1[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x38	TIMx_CCR2	Res.	CCR2[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x3C to 0x4C	Res.																																
0x38	TIMx_CCR2	Res.	CCR2[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Refer to [Table 1 on page 39](#) for the register boundary addresses.

## 17.6 TIM13/14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

### 17.6.1 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		
Res.						CKD[1:0]		ARPE	Res.						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, Tlx),

- 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

### 17.6.2 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
Res.														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

### 17.6.3 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
Res.						CC1OF	Res.							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.

### 17.6.4 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
Res.															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.

### 17.6.5 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
Res.								OC1M[2:0]		OC1PE	OC1FE	CC1S[1:0]			
Res.								IC1F[3:0]			IC1PSC[1:0]				
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

#### Output compare mode

Bits 15:7 Reserved, must be kept at reset value.

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, must be kept at reset value.

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  $f_{DTS}$   
 1000:  $f_{SAMPLING}=f_{DTS}/8$ , N=6  
 0001:  $f_{SAMPLING}=f_{CK\_INT}$ , N=2  
 1001:  $f_{SAMPLING}=f_{DTS}/8$ , N=8  
 0010:  $f_{SAMPLING}=f_{CK\_INT}$ , N=4  
 1010:  $f_{SAMPLING}=f_{DTS}/16$ , N=5  
 0011:  $f_{SAMPLING}=f_{CK\_INT}$ , N=8  
 1011:  $f_{SAMPLING}=f_{DTS}/16$ , N=6  
 0100:  $f_{SAMPLING}=f_{DTS}/2$ , N=6  
 1100:  $f_{SAMPLING}=f_{DTS}/16$ , N=8  
 0101:  $f_{SAMPLING}=f_{DTS}/2$ , N=8  
 1101:  $f_{SAMPLING}=f_{DTS}/32$ , N=5  
 0110:  $f_{SAMPLING}=f_{DTS}/4$ , N=6  
 1110:  $f_{SAMPLING}=f_{DTS}/32$ , N=6  
 0111:  $f_{SAMPLING}=f_{DTS}/4$ , N=8  
 1111:  $f_{SAMPLING}=f_{DTS}/32$ , N=8

*Note:* In current silicon revision,  $f_{DTS}$  is replaced in the formula by  $CK\_INT$  when  $ICx[F3:0]=1, 2$  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).

**17.6.6 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
Res.												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 51. 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.

### 17.6.7 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

### 17.6.8 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.

### 17.6.9 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 [Section 17.4.1: Time-base unit on page 361](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

**17.6.10 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 CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

**17.6.11 TIM14 option register (TIM14\_OR)**

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TI1_RMP[1:0]														
															rw

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

Bit 1:0 **TI1\_RMP[1:0]**: Timer Input 1 remap

Set and cleared by software.

00: TIM14 Channel1 is connected to the GPIO. Refer to the Alternate function mapping table in the device datasheets.

01: the RTC\_CLK is connected to the TIM14\_CH1 input for calibration purposes

10: TIM14\_CH1 input is connected to HSE/32 clock

11: TIM14\_CH1 input is connected to MCO clock.

**17.6.12 TIM13/14 register map**

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

**Table 52. 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	<b>TIMx_CR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKD [1:0]	ARPE	Res.	Res.	Res.	Res.	Res.	URS	UDIS	CEN		
	Reset value																							0	0	0					0	0	0		
0x08	<b>TIMx_SMCR</b>	Not Available																																	
	Reset value																																		
0x0C	<b>TIMx_DIER</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																	0	0
0x10	<b>TIMx_SR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	CC1IF	UIF							
	Reset value																								0									0	0
0x14	<b>TIMx_EGR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																																	0	0



Table 52. 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
0x18	TIMx_CCMR1 Output compare mode	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]																									
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Res.	IC1F [3:0]			IC1PSC [1:0]	CC1S [1:0]																										
	Reset value																										0	0	0	0	0	0	0
0x1C	Res.																																
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																
0x24	TIMx_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																
0x30	Res.																																
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																
0x38	TIM14_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																
0x3C to 0x4C	Res.																																



## 18 General-purpose timers (TIM15/16/17)

### 18.1 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. The TIM15 can be synchronized with other timers.

### 18.2 TIM15 main features

TIM15 includes the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

## 18.3 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 122. TIM15 block diagram

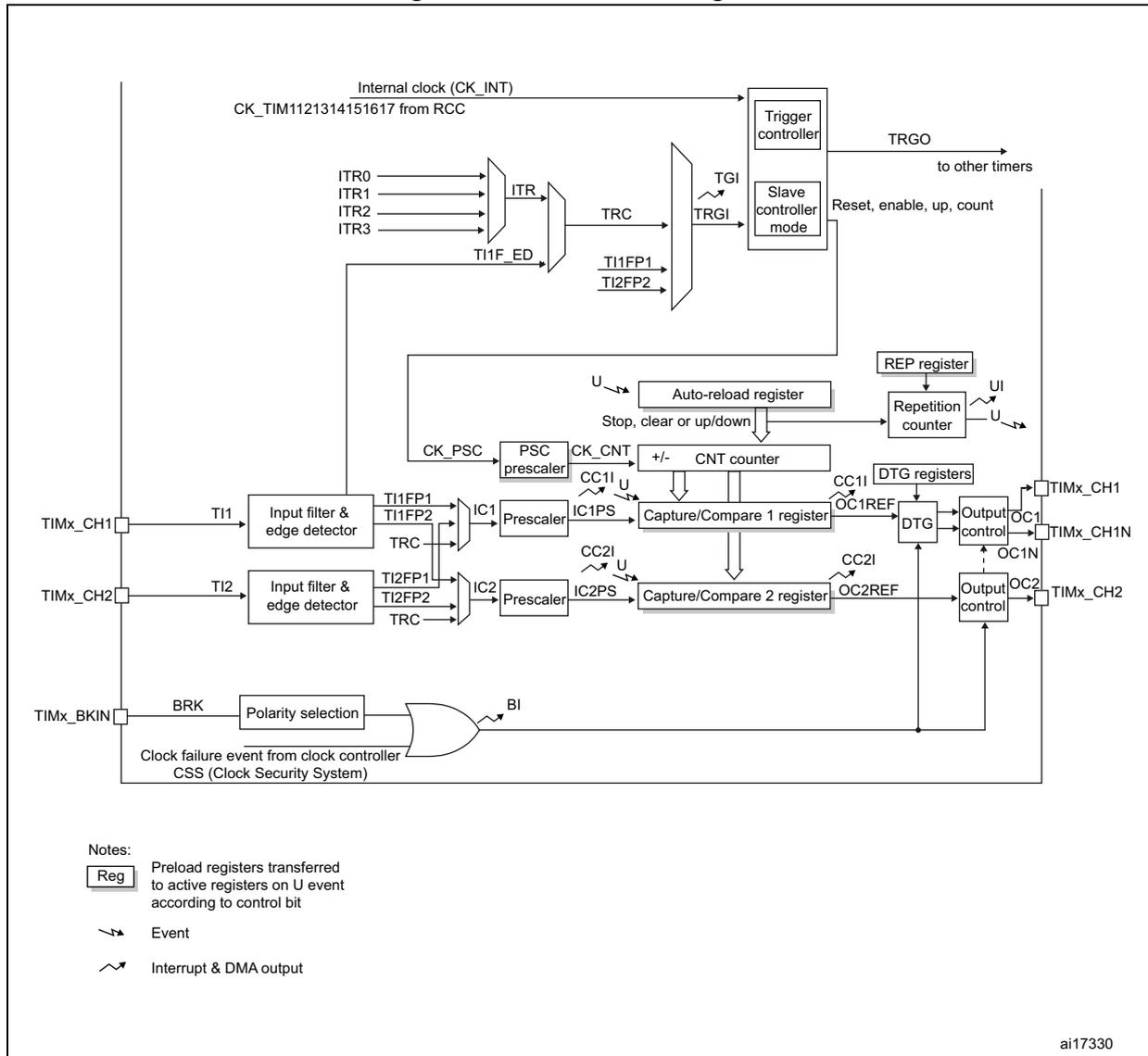
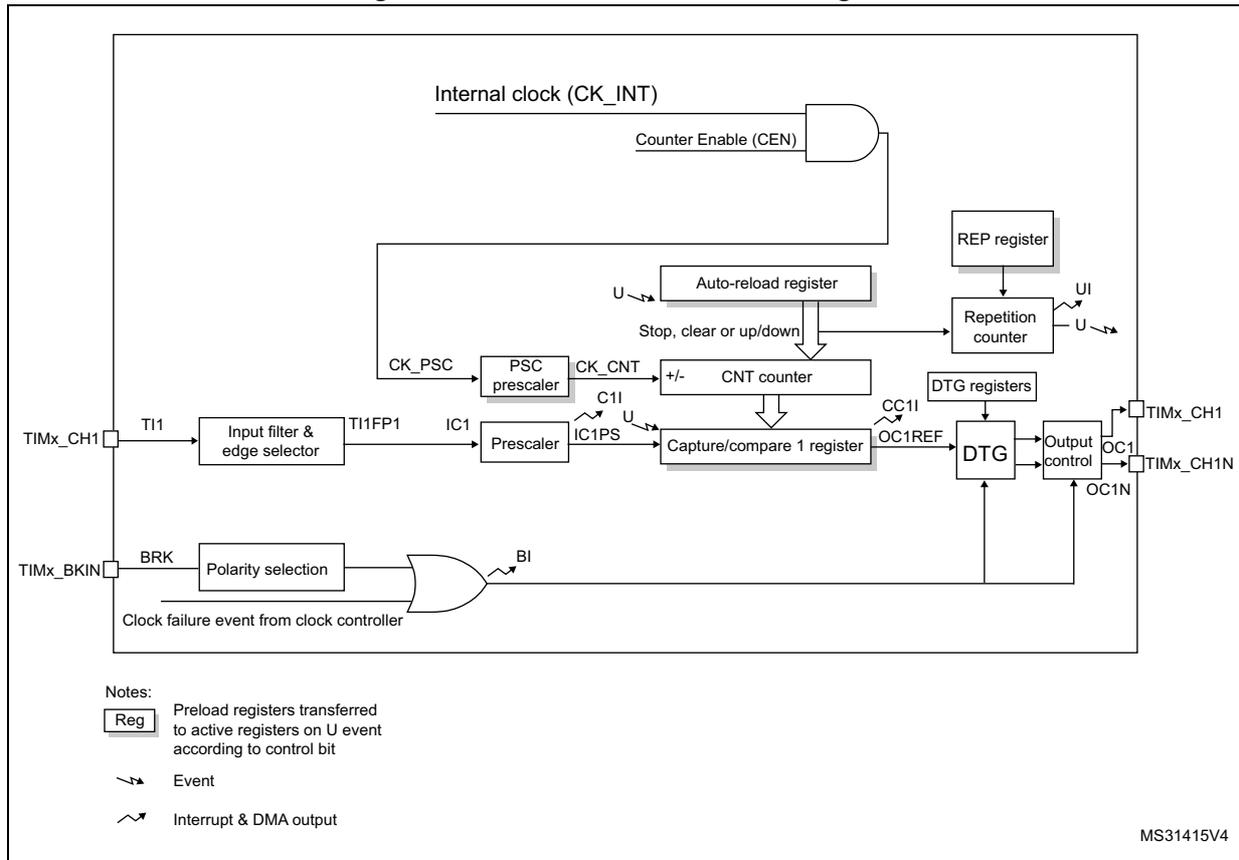


Figure 123. TIM16 and TIM17 block diagram



## 18.4 TIM15/16/17 functional description

### 18.4.1 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 102* and *Figure 103* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 124. Counter timing diagram with prescaler division change from 1 to 2

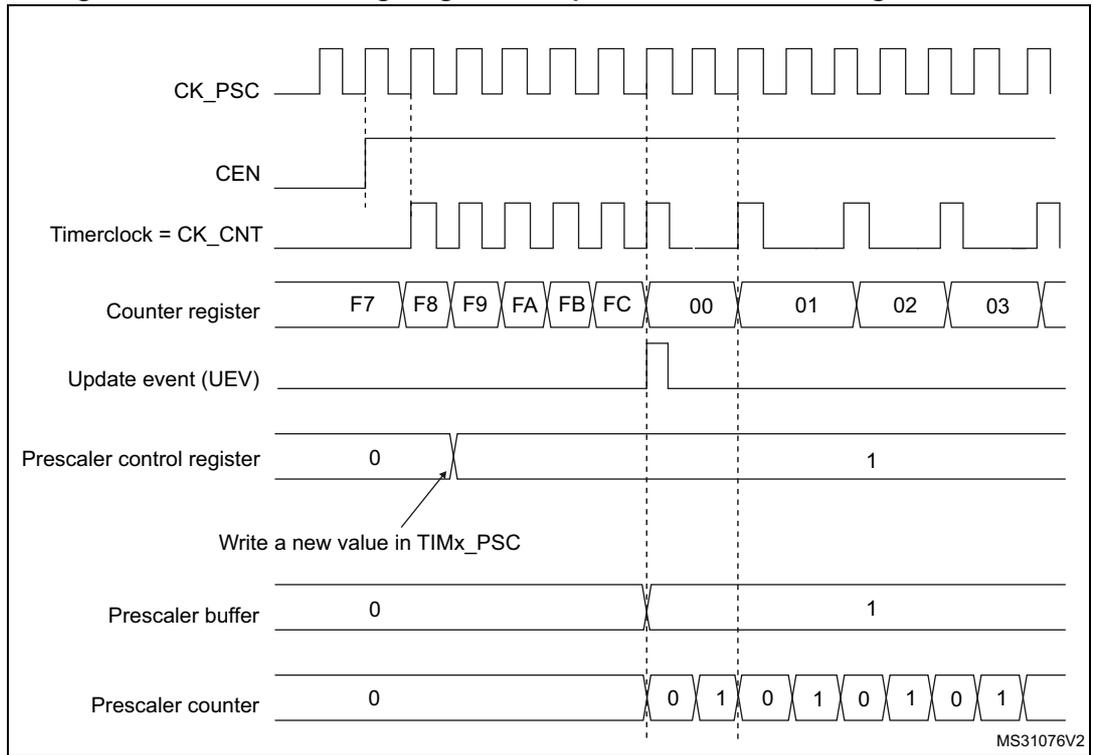
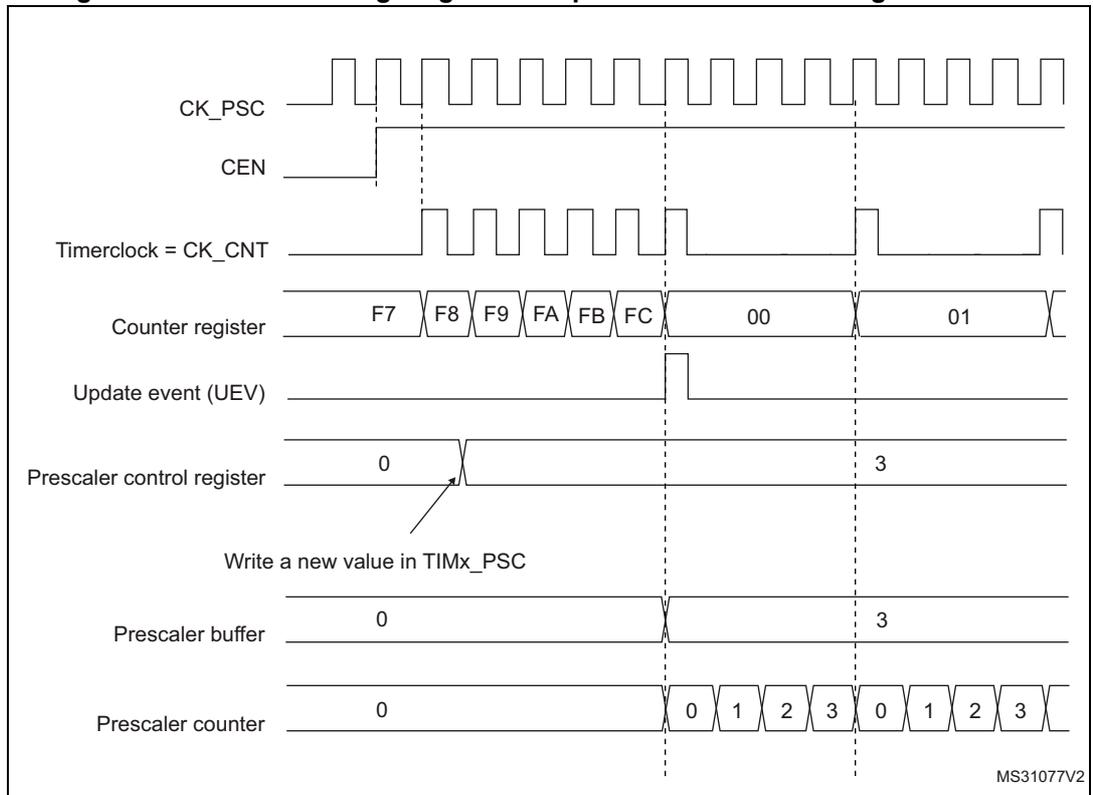


Figure 125. Counter timing diagram with prescaler division change from 1 to 4



## 18.4.2 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.

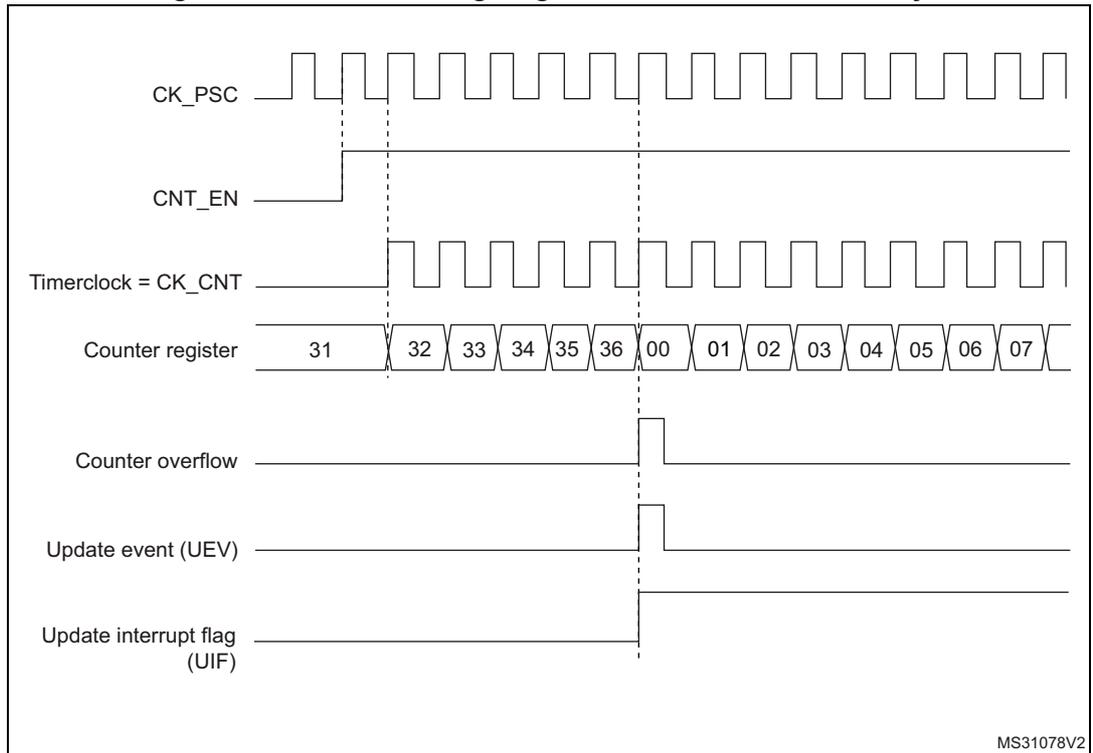
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as 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).

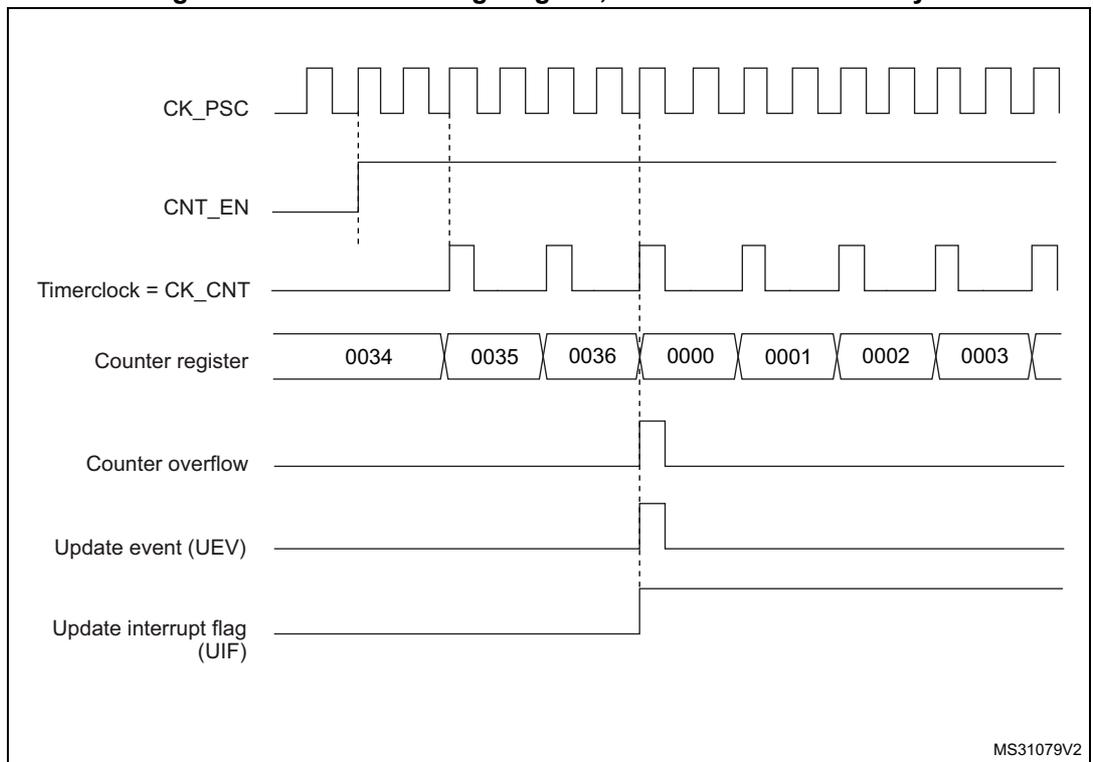
The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

Figure 126. Counter timing diagram, internal clock divided by 1



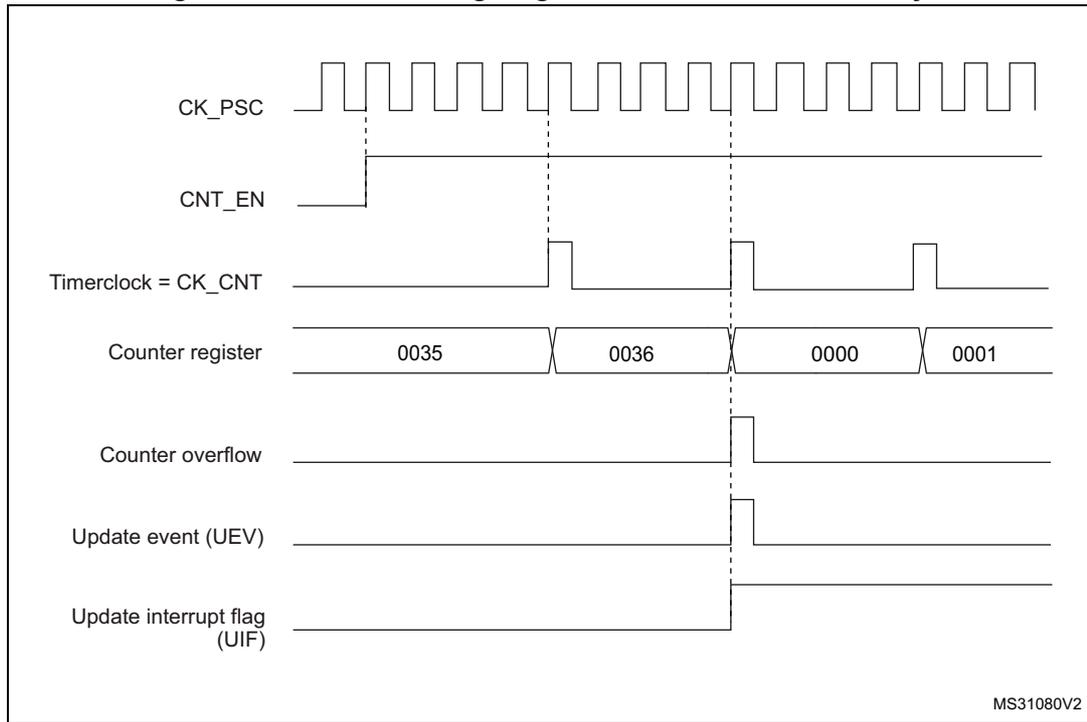
MS31078V2

Figure 127. Counter timing diagram, internal clock divided by 2



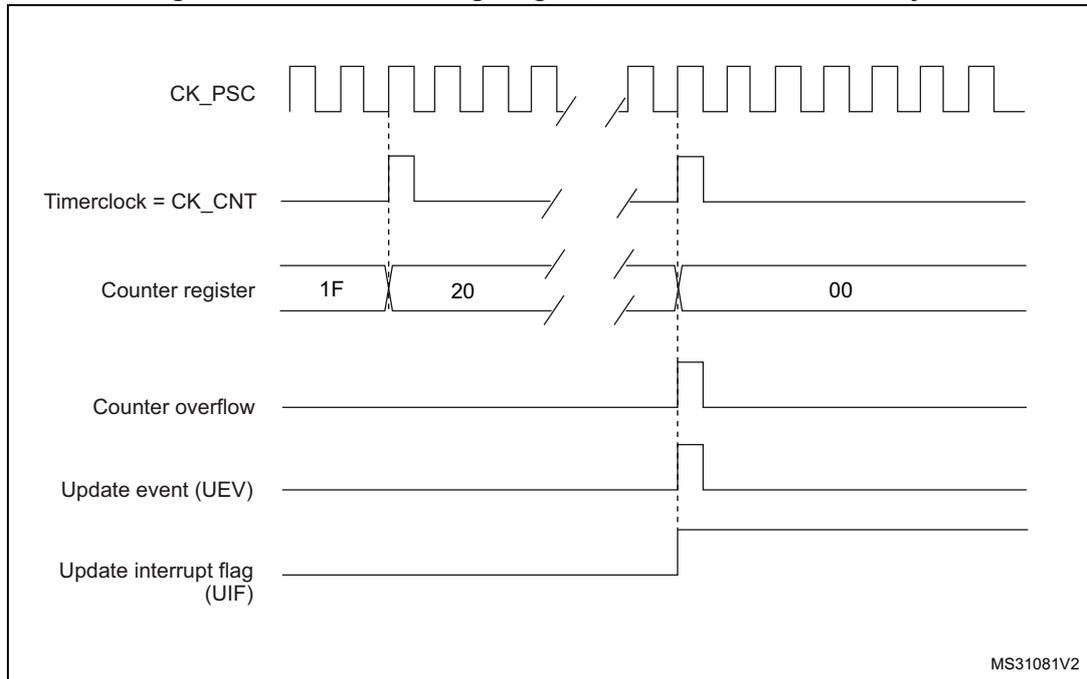
MS31079V2

**Figure 128. Counter timing diagram, internal clock divided by 4**



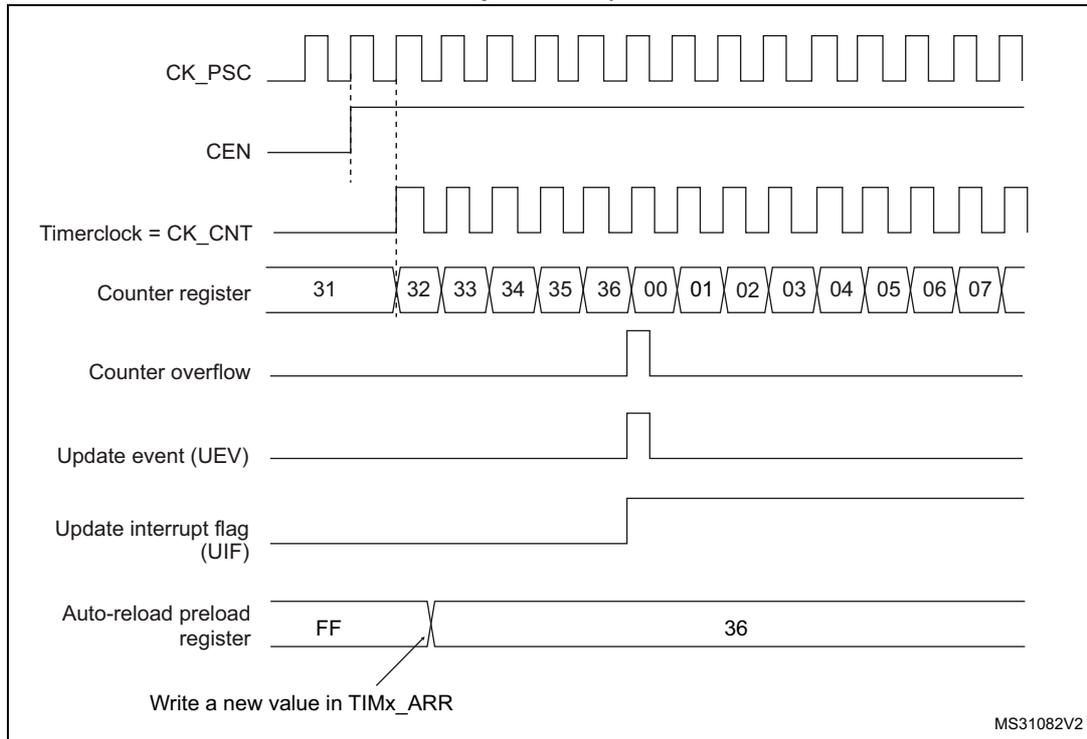
MS31080V2

**Figure 129. Counter timing diagram, internal clock divided by N**

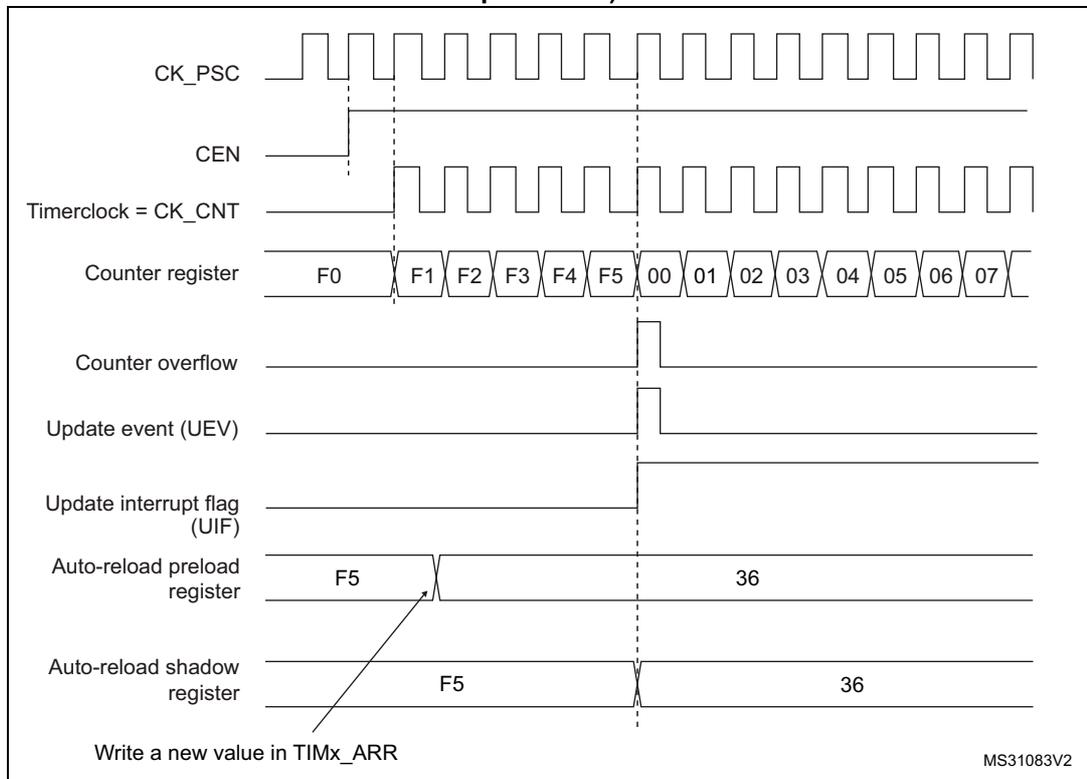


MS31081V2

**Figure 130. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 131. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 18.4.3 Repetition counter

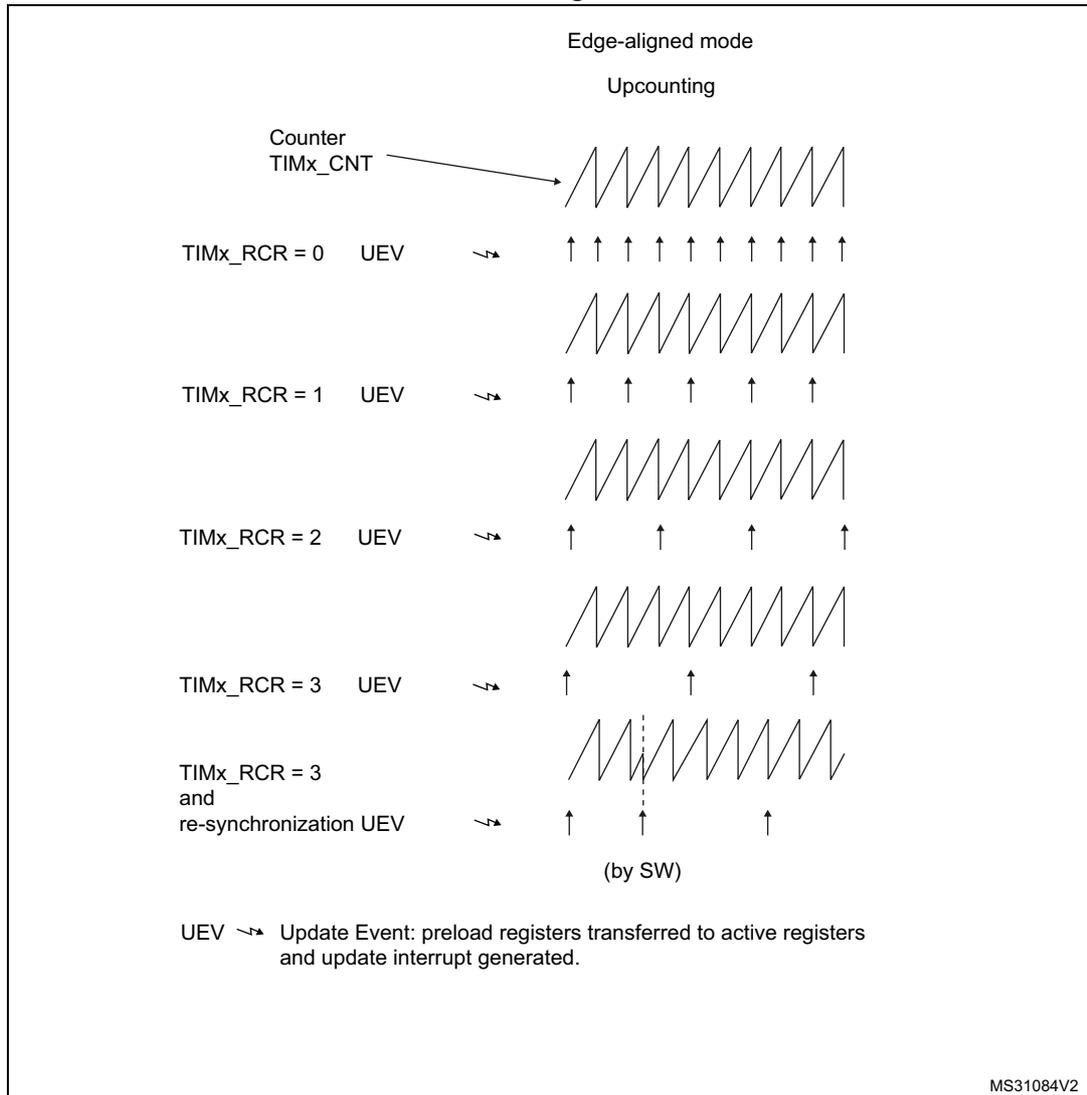
*Section 17.4.1: Time-base unit* describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

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 132*). 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 132. Update rate examples depending on mode and TIMx\_RCR register settings**



### 18.4.4 Clock selection

The counter clock can be provided by the following clock sources:

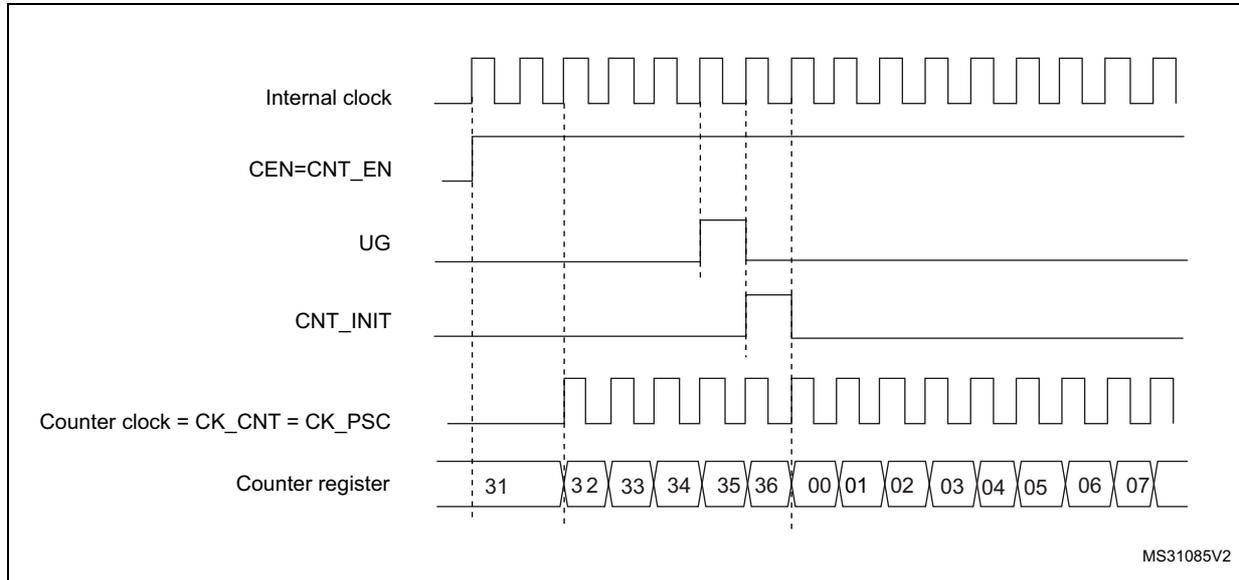
- Internal clock (CK\_INT)
- External clock mode1: external input pin (only for TIM15)
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, you can configure TIM2 to act as a prescaler for TIM15. Refer to [Using one timer as prescaler for another](#) for more details.

#### Internal clock source (CK\_INT)

For TIM15, 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 109 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

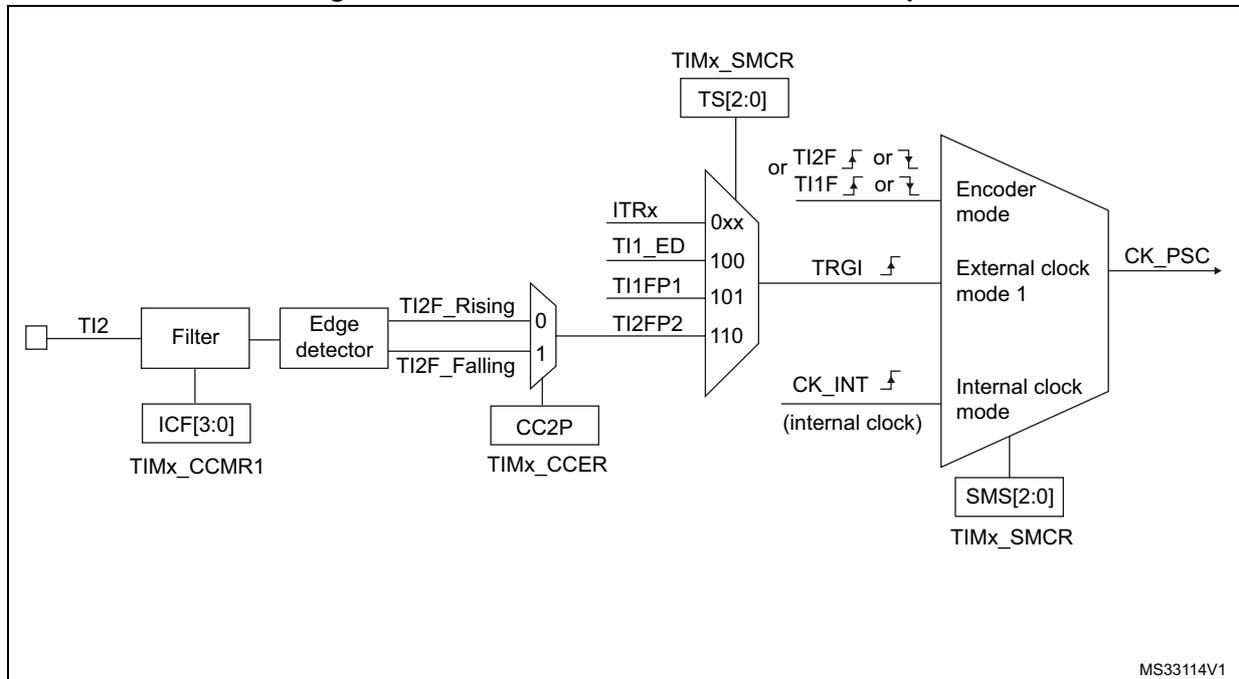
Figure 133. Control circuit in normal mode, internal clock divided by 1



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 134. 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:

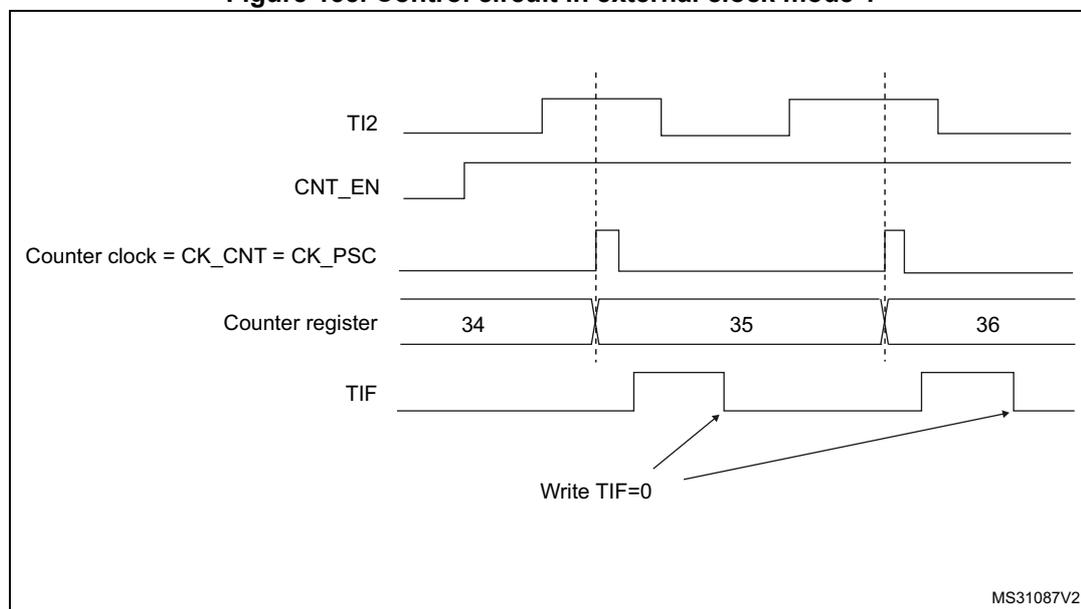
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 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 135. Control circuit in external clock mode 1**



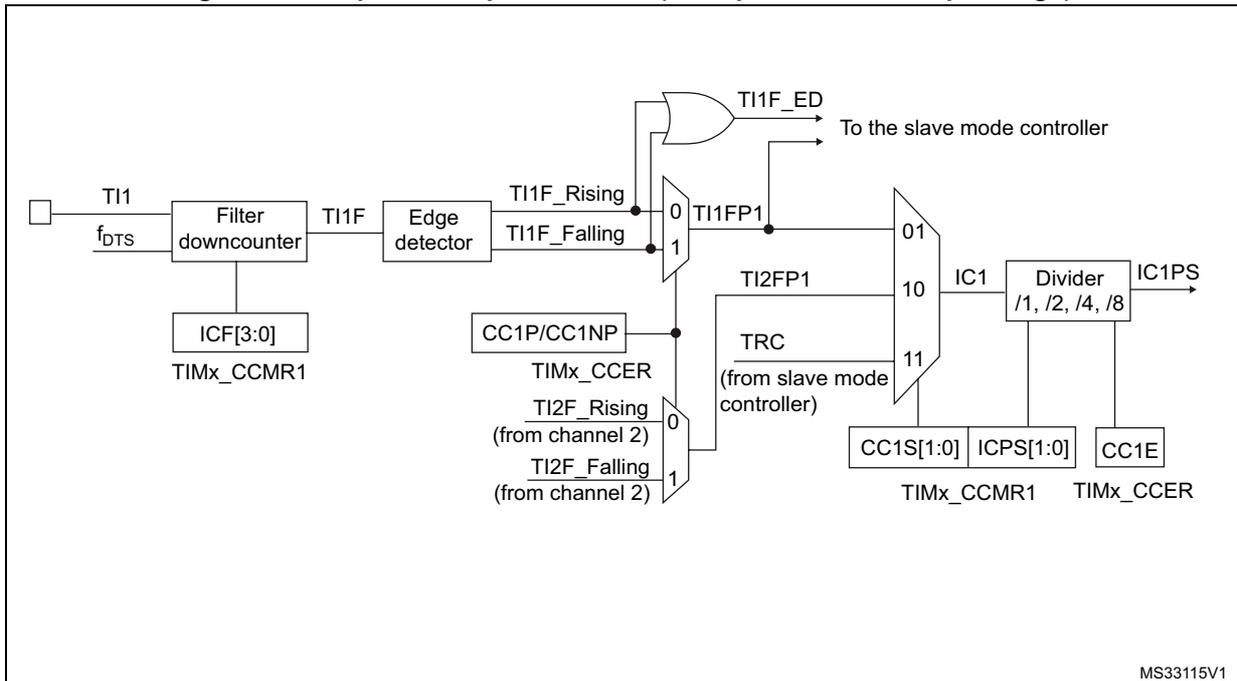
### 18.4.5 Capture/compare channels

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 112](#) to [Figure 139](#) 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 136. 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 137. Capture/compare channel 1 main circuit

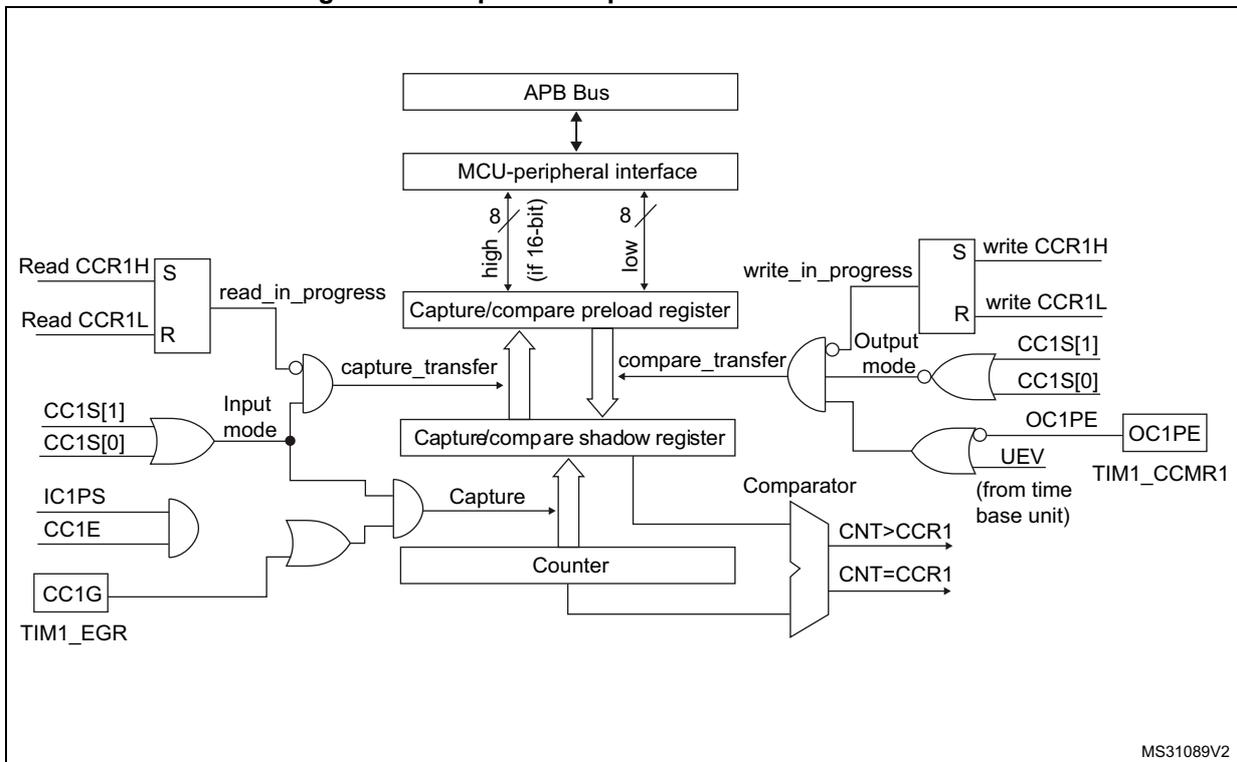
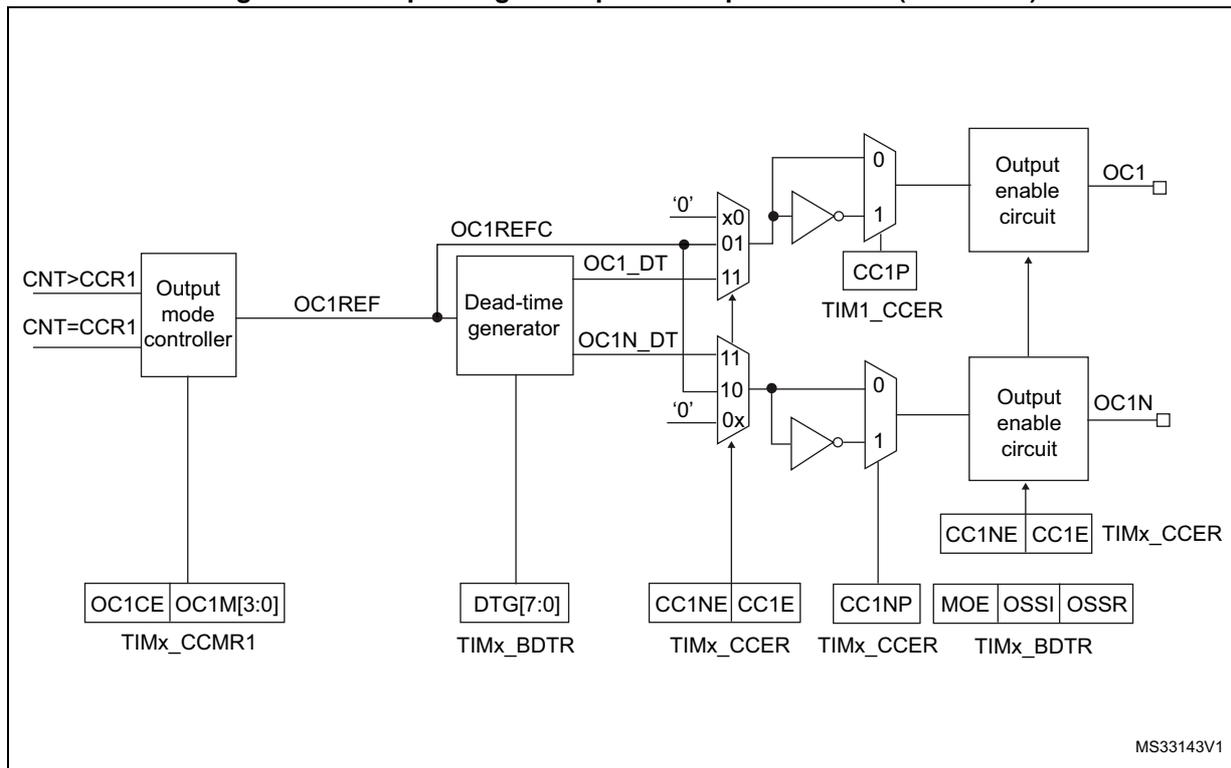
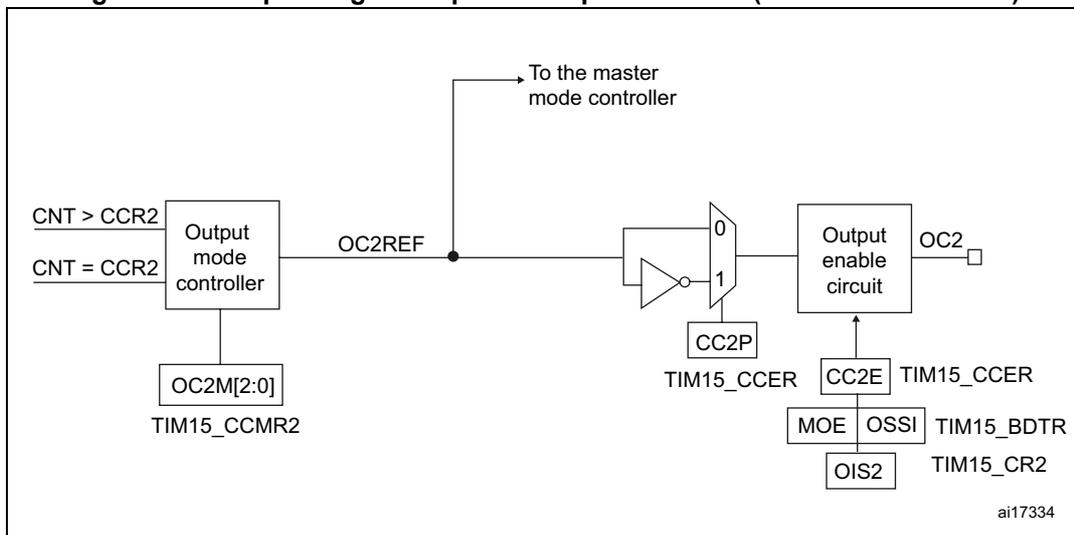


Figure 138. Output stage of capture/compare channel (channel 1)



MS33143V1

Figure 139. Output stage of capture/compare channel (channel 2 for TIM15)



ai17334

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.

### 18.4.6 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:

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 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 most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.
3. 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).
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, 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.

### 18.4.7 PWM input mode (only for TIM15)

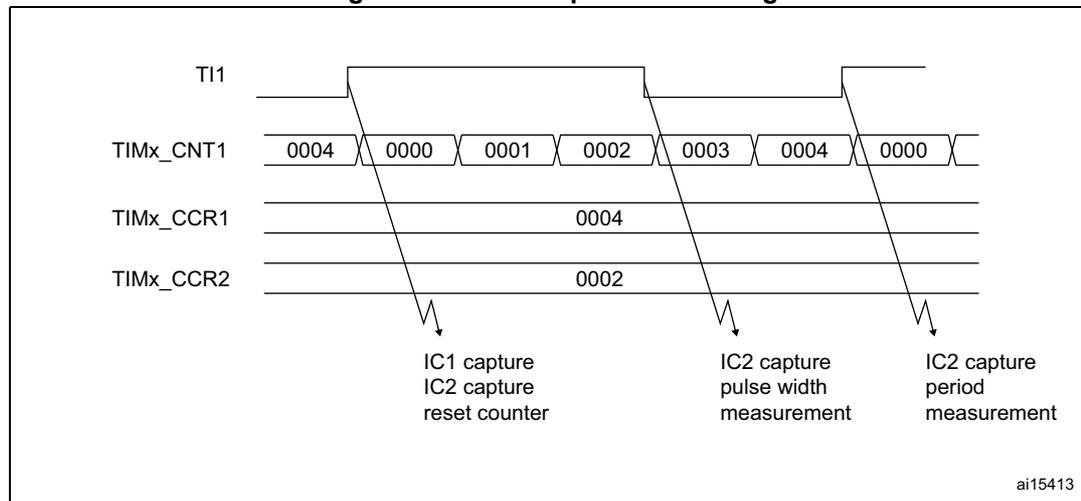
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same T1x input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two T1xFP 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): write the CC1P bit to '0' (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): write the CC2P bit to '1' (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 140. 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.

### 18.4.8 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.

### 18.4.9 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:

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).
4. 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.

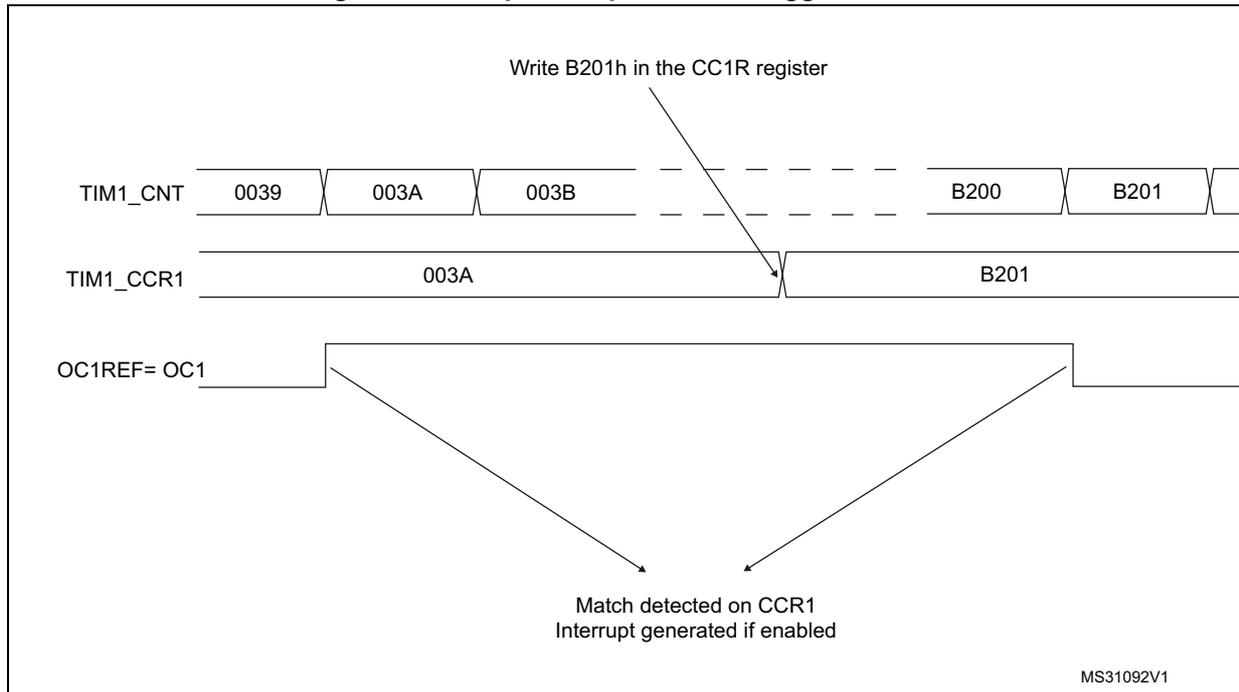
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 116](#).

Figure 141. Output compare mode, toggle on OC1



### 18.4.10 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  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  (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.

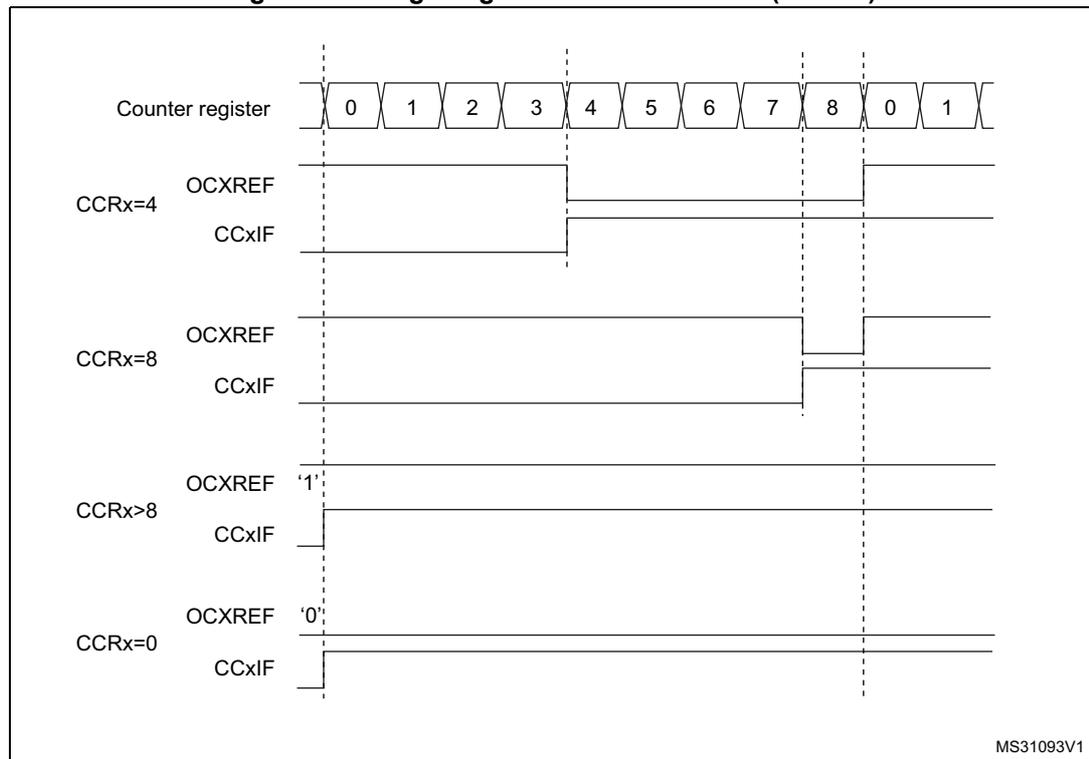
**PWM edge-aligned mode**

- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to the [Upcounting mode on page 363](#).

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 117](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 142. 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 416](#)

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.

**18.4.11 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 54: Output control bits for complementary OCx and OCxN channels with break feature on page 450](#) for more details. In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

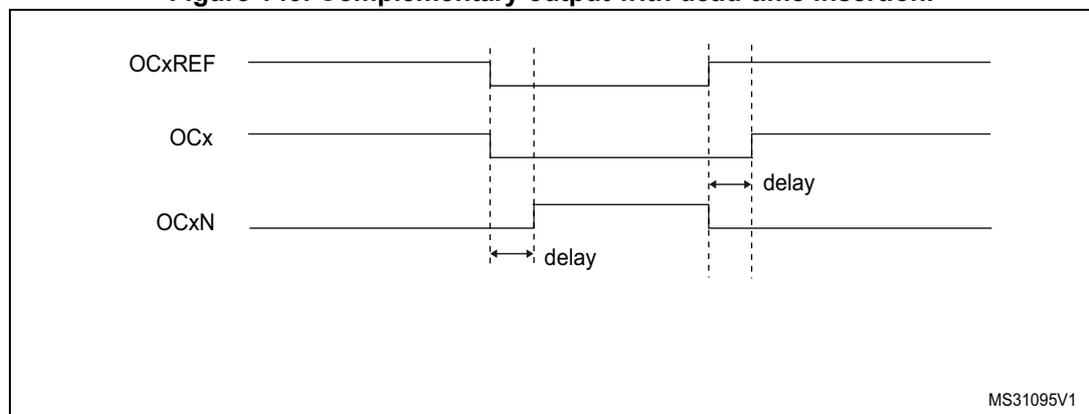
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

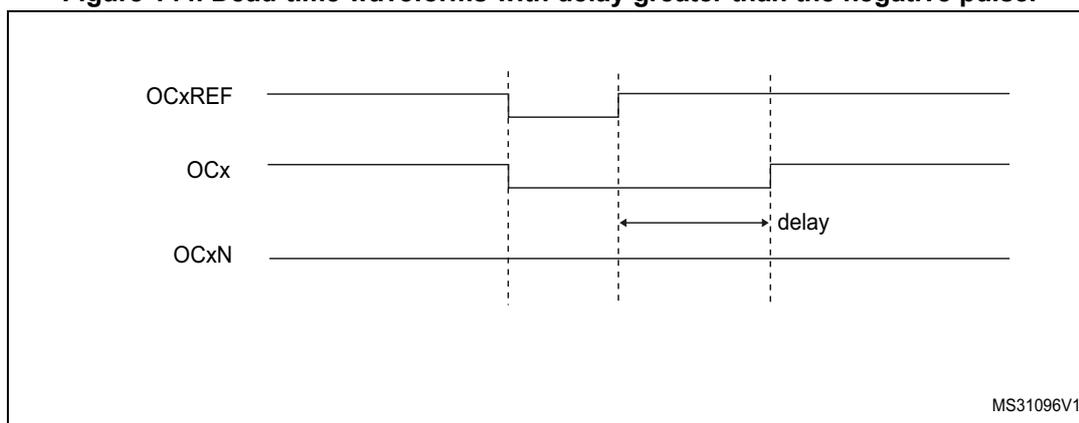
If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

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 143. Complementary output with dead-time insertion.**

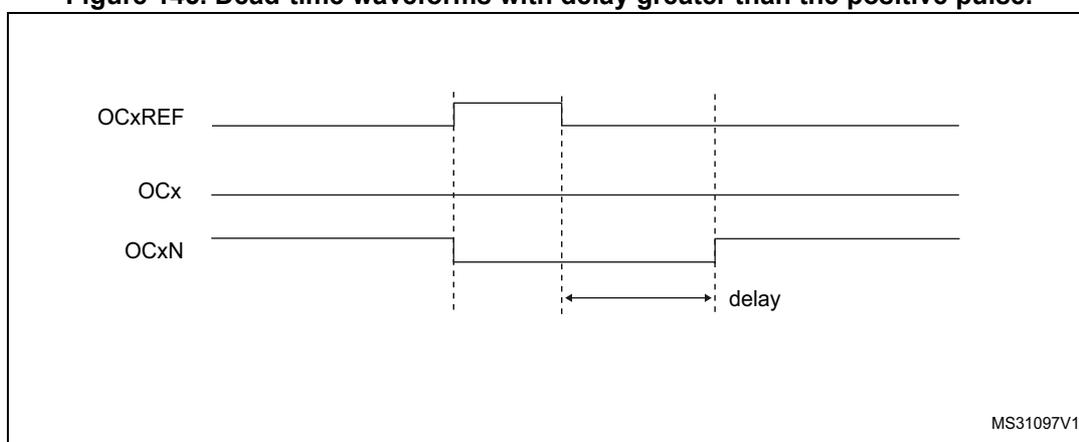


**Figure 144. Dead-time waveforms with delay greater than the negative pulse.**



MS31096V1

**Figure 145. Dead-time waveforms with delay greater than the positive pulse.**



MS31097V1

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 18.5.15: TIM15 break and dead-time register \(TIM15\\_BDTR\) on page 453](#) 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.

### 18.4.12 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 54: Output control bits for complementary OCx and OCxN channels with break feature on page 450](#) 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 7.2.7: 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.

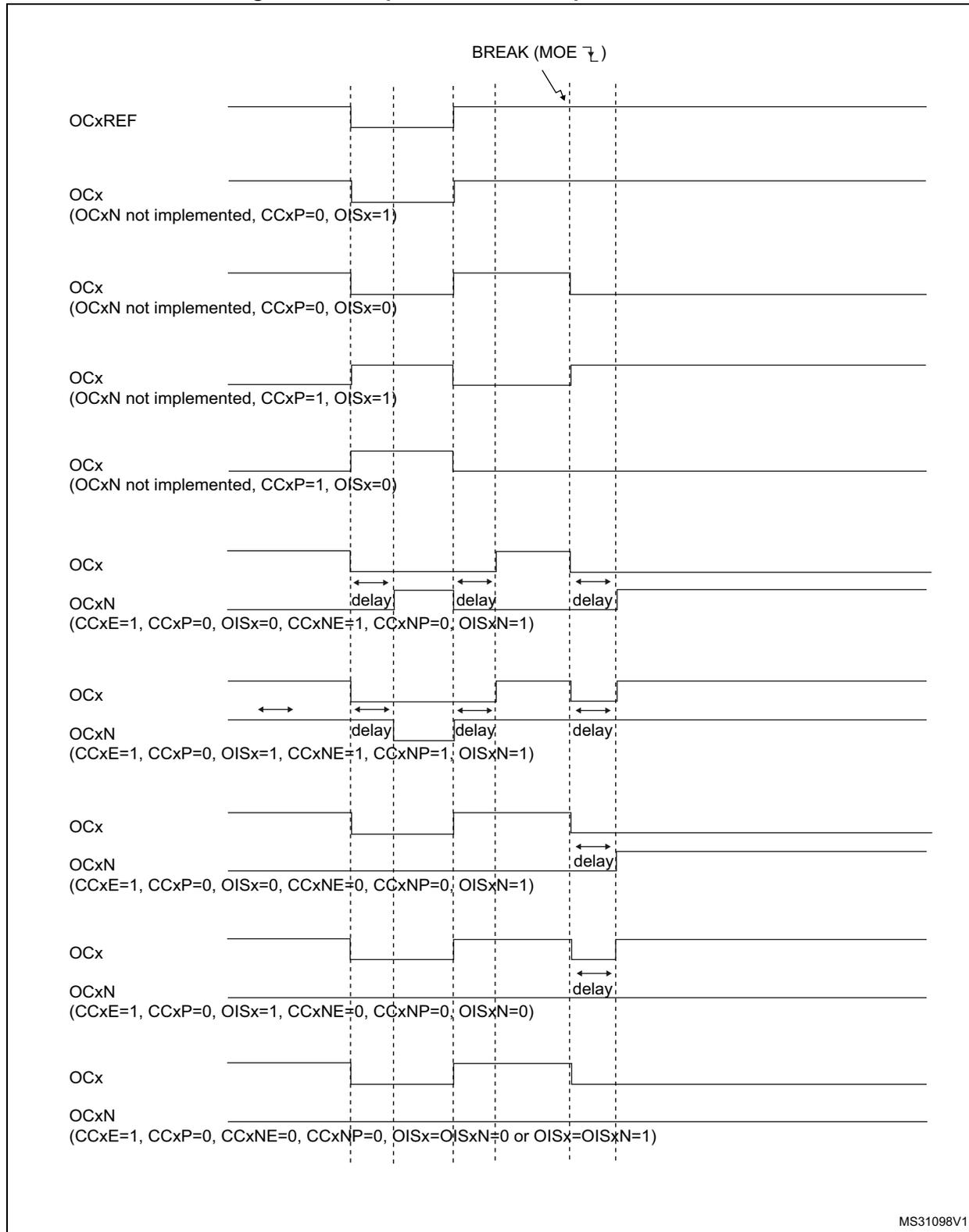
*Note:* The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

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 18.5.15: TIM15 break and dead-time register \(TIM15\\_BDTR\) on page 453](#). The LOCK bits can be written only once after an MCU reset.

The [Figure 146](#) shows an example of behavior of the outputs in response to a break.

Figure 146. Output behavior in response to a break.



### 18.4.13 One-pulse mode

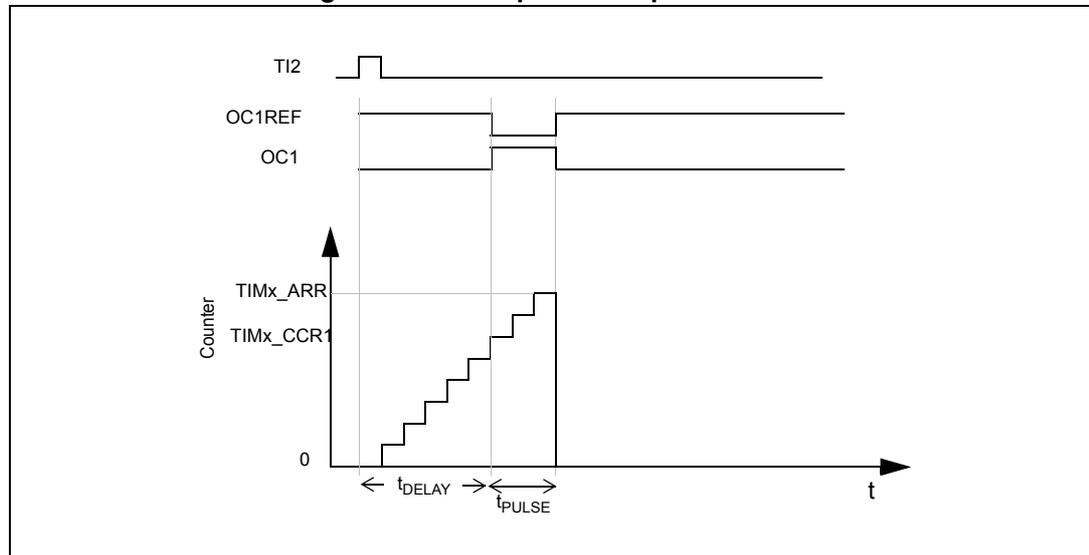
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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:  $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ )
- In downcounting:  $CNT > CCRx$

**Figure 147. 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:

1. Map TI2FP2 to TI2 by writing  $CC2S='01'$  in the TIMx\_CCMR1 register.
2. TI2FP2 must detect a rising edge, write  $CC2P='0'$  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  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say 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 Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY min}}$  we can get.

If 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.

### 18.4.14 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.

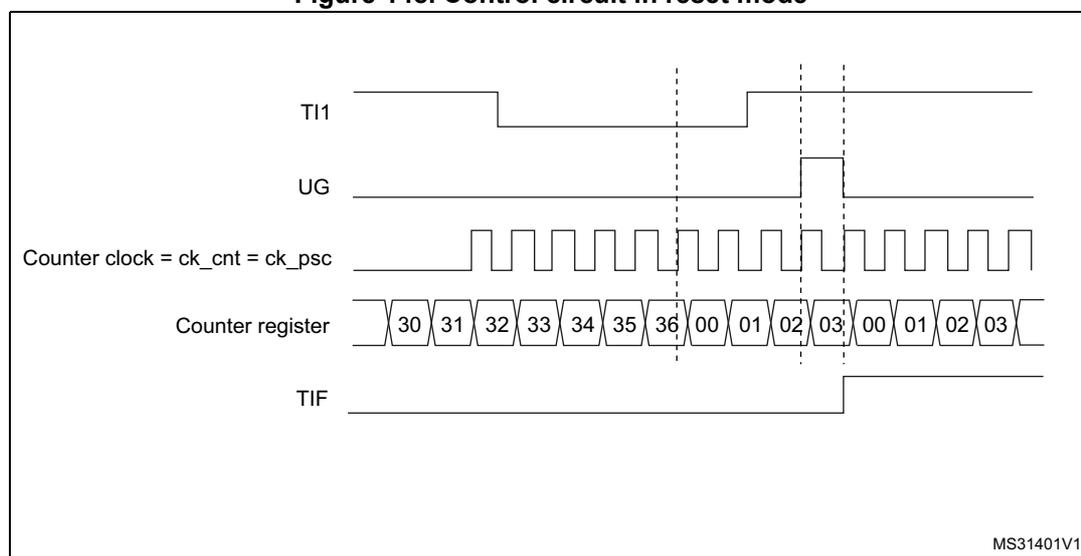
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

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. Write CC1P=0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=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, 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 148. Control circuit in reset mode



**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

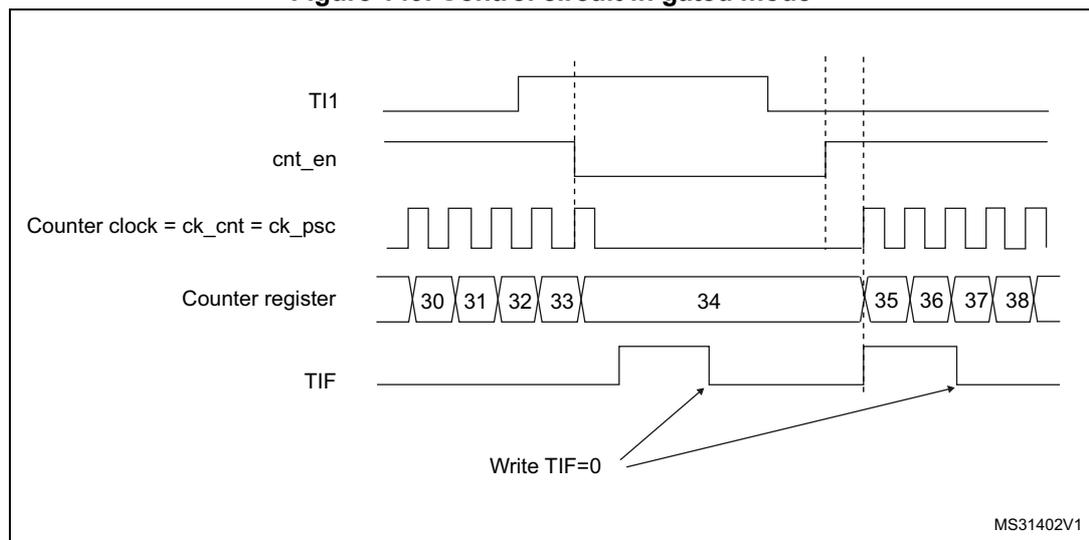
In the following example, the upcounter counts only when TI1 input is low:

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. Write CC1P=1 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 149. Control circuit in gated mode**



### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

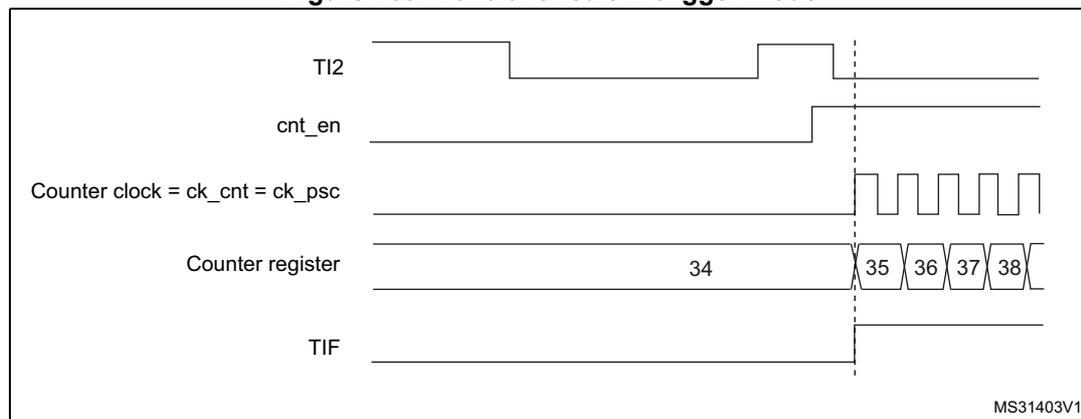
In the following example, the upcounter starts in response to a rising edge on TI2 input:

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. Write CC2P=1 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 150. Control circuit in trigger mode**



### 18.4.15 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining. Refer to [Section 16.3.15: Timer synchronization on page 329](#) for details.

### 18.4.16 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M4 with FPU 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 31.16.2: Debug support for timers, watchdog, bxCAN and I2C](#).

## 18.5 TIM15 registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

### 18.5.1 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	
Res.						CKD[1:0]		ARPE	Res.				OPM	URS	UDIS	CEN
						rw		rw					rw	rw	rw	rw

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

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) 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, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 18.5.2 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
Res.				OIS2	OIS1N	OIS1	Res.	MMS[2:0]			CCDS	CCUS	Res.	CCPC	
				r/w	r/w	r/w		r/w	r/w	r/w	r/w	r/w		r/w	

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

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, must be kept at reset value.

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, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

*Note: This bit acts only on channels that have a complementary output.*

### 18.5.3 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	
Res.								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 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: T11 Edge Detector (T11F\_ED)

101: Filtered Timer Input 1 (TI1FP1)

110: Filtered Timer Input 2 (TI2FP2)

See [Table 53: TIMx Internal trigger connection on page 441](#) 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 T11F\_ED is selected as the trigger input (TS='100'). Indeed, T11F\_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.*

Table 53. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 000) <sup>(1)</sup>	ITR1 (TS = 001) <sup>(1)</sup>	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM15	TIM2	TIM3	TIM16_OC1	TIM17_OC1

1. ITR0 and ITR1 triggers available only in high density value line devices.

**18.5.4 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	Res.			CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.			CC2IE	CC1IE	UIE
	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

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

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, 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

### 18.5.5 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	
Res.					CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.			CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0					rc_w0	rc_w0	rc_w0

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

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag  
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
0: No overcapture has been detected  
1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag  
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.  
0: No break event occurred  
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag  
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is 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, must be kept at reset value.

- Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag  
refer to CC1IF description
- Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag  
**If channel CC1 is configured as output:**  
 This flag is set by hardware when the counter matches the compare value, 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 18.5.3: TIM15 slave mode control register \(TIM15\\_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 18.5.6 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	
Res.								BG	TG	COMG	Res.			CC2G	CC1G	UG
								w	w	rw				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 the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). 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).

### 18.5.7 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]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

**Output compare mode:**

Bit 15 Reserved, must be kept at reset value.

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, must be kept at reset value.

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 15:12 **IC2F**: Input capture 2 filterBits 11:10 **IC2PSC[1:0]**: Input capture 2 prescalerBits 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).

### 18.5.8 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
	Res.								CC1NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
									r/w		r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity  
refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity  
refer to CC1P description

- Bit 4 **CC2E**: Capture/Compare 2 output enable  
refer to CC1E description
- Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity  
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 54. Output control bits for complementary OCx and OCxN channels with break feature**

Control bits					Output states <sup>(1)</sup>		
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	X	0	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	1			
			1	0			
			1	1			
			1	0	0	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			
			1	0			
			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.*



### 18.5.9 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

### 18.5.10 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”).

### 18.5.11 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.

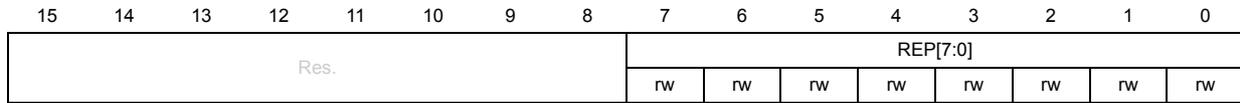
Refer to the [Section 17.4.1: Time-base unit on page 361](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 18.5.12 TIM15 repetition counter register (TIM15\_RCR)

Address offset: 0x30

Reset value: 0x0000



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.

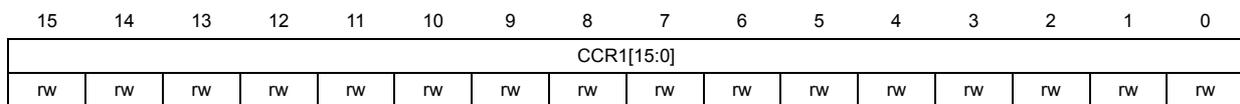
Each time the REP\_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP\_CNT is reloaded with REP value only at the repetition update event U\_RC, any write to the TIMx\_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

### 18.5.13 TIM15 capture/compare register 1 (TIM15\_CCR1)

Address offset: 0x34

Reset value: 0x0000



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).

### 18.5.14 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).

### 18.5.15 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 18.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 448](#)).

**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 18.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 448](#)).

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 18.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 448](#)).

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<sub>dtg</sub> with t<sub>dtg</sub>=t<sub>DTS</sub>

DTG[7:5]=10x => DT=(64+DTG[5:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=2x t<sub>DTS</sub>

DTG[7:5]=110 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=8x t<sub>DTS</sub>

DTG[7:5]=111 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=16x t<sub>DTS</sub>

Example if T<sub>DTS</sub>=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).*

### 18.5.16 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
Res.		DBL[4:0]					Res.		DBA[4:0]						
Res.		rw	rw	rw	rw	rw	Res.		rw	rw	rw	rw	rw		

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-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,
- ...

### 18.5.17 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  $(\text{TIMx\_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

18.5.18 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 55. 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	Res.	CKD [1:0]	0	0	ARPE	0	Res.		OPM	URS	UDIS	CEN																							
	Reset value																								0	0	0			0	0	0	0			
0x04	TIM15_CR2	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]		CCDS	CCUS	Res.	CCPC																								
	Reset value																							0	0	0	0	0	0	0	0	0	0			
0x08	TIM15_SMCR	Res.	Res.	MSM	TS[2:0]		Res.	SMS[2:0]																												
	Reset value																										0	0	0	0	0	0	0			
0x0C	TIM15_DIER	Res.	0	TDE	Res.	Res.	Res.	Res.	OC2DE	OC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE																	
	Reset value																		0						0	0	0	0	0	0	0	0	0	0		
0x10	TIM15_SR	Res.	Res.	Res.	BIF	TIF	COMIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF																							
	Reset value																										0	0	0	0	0	0	0	0	0	
0x14	TIM15_EGR	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	Res.	CC2G	CC1G	UG																							
	Reset value																											0	0	0			0	0	0	
0x18	TIM15_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																			
	TIM15_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
Reset value																																				
0x20	TIM15_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																			
0x24	TIM15_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																			
0x28	TIM15_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																							
	Reset value																																			



Table 55. 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	Res.	ARR[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TIM15_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]																							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIM15_CCR1	Res.	CCR1[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x38	TIM15_CCR2	Res.	CCR2[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x44	TIM15_BDTR	Res.	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	
0x48	TIM15_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																
	Reset value																																
0x4C	TIM15_DMAR	Res.	DMAB[15:0]																														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Refer to [Section 2.2.2 on page 39](#) for the register boundary addresses.

## 18.6 TIM16&TIM17 registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

### 18.6.1 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	
Res.						CKD[1:0]		ARPE	Res.				OPM	URS	UDIS	CEN
						rw		rw					rw	rw	rw	rw

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

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) 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, must be kept at reset value.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled.

These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

### 18.6.2 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		
Res.						OIS1N	OIS1	Res.						CCDS	CCUS	Res.	CCPC
						r/w	r/w							r/w	r/w		r/w

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

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BKR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BKR register).*

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

*Note: This bit acts only on channels that have a complementary output.*

### 18.6.3 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.	Res.	Res.			CC1DE	UDE	BIE	Res	COMIE	Res.			CC1IE	UIE	
						rw	rw	rw		rw			rw	rw	

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, always read as 0.

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

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable

0: CC1 DMA request disabled

1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled

1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable

0: Break interrupt disabled

1: Break interrupt enabled

Bit 6 Reserved, always read as 0.

Bit 5 **COMIE**: COM interrupt enable

0: COM interrupt disabled

1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

**18.6.4 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
Res.						CC1OF	Res.	BIF	Res	COMIF	Res.			CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, always read as 0.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

**If channel CC1 is configured as output:**

This flag is set by hardware when the counter matches the compare value, 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.

### 18.6.5 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
Res.								BG	Res.	COMG	Res.			CC1G	UG
								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 Reserved, always read as 0.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). 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).

### 18.6.6 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
Res.								Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								IC1F[3:0]				IC1PSC[1:0]			
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

**Output compare mode:**

Bits 15:7 Reserved, must be kept at reset value.

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, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N 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).*

### 18.6.7 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
Res.												CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

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 TlxFP1's rising edge TlxFP1 is not inverted.

01: Inverted/falling edge: circuit is sensitive to TlxFP1's falling edge, TlxFP1 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 TlxFP1, TlxFP1 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 56. Output control bits for complementary OCx and OCxN channels with break feature**

Control bits					Output states <sup>(1)</sup>	
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	X	0	0	0	Output Disabled (not driven by the timer)	
		0	0	1	Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0	
		0	1	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	1	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	0	0	Off-State (output enabled with inactive state)	
		1	0	1	Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1	
		1	1	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	
		1	1	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. 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.

### 18.6.8 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

### 18.6.9 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”).

### 18.6.10 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.

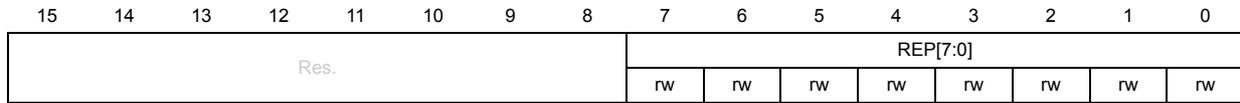
Refer to the [Section 17.4.1: Time-base unit on page 361](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 18.6.11 TIM16&TIM17 repetition counter register (TIMx\_RCR)

Address offset: 0x30

Reset value: 0x0000



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.

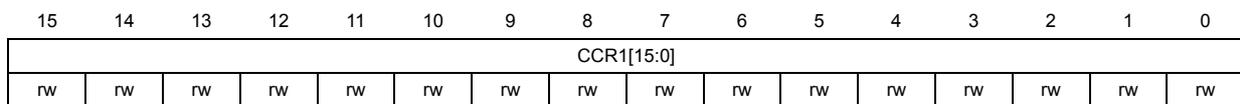
Each time the REP\_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP\_CNT is reloaded with REP value only at the repetition update event U\_RC, any write to the TIMx\_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

### 18.6.12 TIM16&TIM17 capture/compare register 1 (TIMx\_CCR1)

Address offset: 0x34

Reset value: 0x0000



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).

### 18.6.13 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 18.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 448](#)).

**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 18.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 448](#)).

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 18.5.8: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 448](#)).

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<sub>dtg</sub> with t<sub>dtg</sub>=t<sub>DTS</sub>

DTG[7:5]=10x => DT=(64+DTG[5:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=2x t<sub>DTS</sub>

DTG[7:5]=110 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=8x t<sub>DTS</sub>

DTG[7:5]=111 => DT=(32+DTG[4:0])x t<sub>dtg</sub> with T<sub>dtg</sub>=16x t<sub>DTS</sub>

Example if T<sub>DTS</sub>=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).*

### 18.6.14 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				
Res.					DBL[4:0]					Res.					DBA[4:0]				
					rw	rw	rw	rw	rw						rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

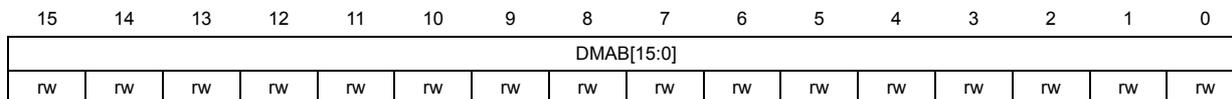
- 00000: TIMx\_CR1,
- 00001: TIMx\_CR2,
- 00010: TIMx\_SMCR,
- ...

**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.

### 18.6.15 TIM16&TIM17 DMA address for full transfer (TIMx\_DMAR)

Address offset: 0x4C

Reset value: 0x0000



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.

### 18.6.16 TIM16&TIM17 register map

TIM16&TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 57. 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	Res.	CKD [1:0]	ARPE	Res.	Res.	Res.	Res.	OPM	URS	UDIS	CEN																						
	Reset value																							0	0	0				0	0	0	0	
0x04	TIMx_CR2	Res.	OISIN	OIS1	TI1S	MMS[2:0]				CCDS	CCUS	Res.	CCPC																					
	Reset value																							0	0	0	0	0	0	0	0	0	0	0
0x0C	TIMx_DIER	Res.	CC1DE	UDE	BIE	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE																						
	Reset value																							0	0	0						0	0	
0x10	TIMx_SR	Res.	CC1OF	Res.	BIF	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF																						
	Reset value																							0	0	0						0	0	
0x14	TIMx_EGR	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	Res.	CC1G	UG																						
	Reset value																									0	0					0	0	



Table 57. 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				
0x18	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	OC1M [2:0]		OC1PE	OC1FE	CC1S [1:0]																												
	Reset value																										0	0	0	0	0	0	0				
0x18	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	IC1F [3:0]			IC1PSC [1:0]	CC1S [1:0]																												
	Reset value																										0	0	0	0	0	0	0	0			
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																										
	Reset value																																				
0x24	TIMx_CNT	Res.	Res.	Res.	CNT[15:0]																																
	Reset value																																				
0x28	TIMx_PSC	Res.	Res.	Res.	PSC[15:0]																																
	Reset value																																				
0x2C	TIMx_ARR	Res.	Res.	Res.	ARR[15:0]																																
	Reset value																																				
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																				
0x34	TIMx_CCR1	Res.	Res.	Res.	CCR1[15:0]																																
	Reset value																																				
0x44	TIMx_BDTR	Res.	MOE	AOE	BKP	BKE	OSSR	LOCK [1:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																			
	Reset value																																				
0x48	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																				
0x4C	TIMx_DMAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																									
	Reset value																																				

Refer to [Section 2.2.2 on page 39](#) for the register boundary addresses.

## 19 Infrared interface (IRTIM)

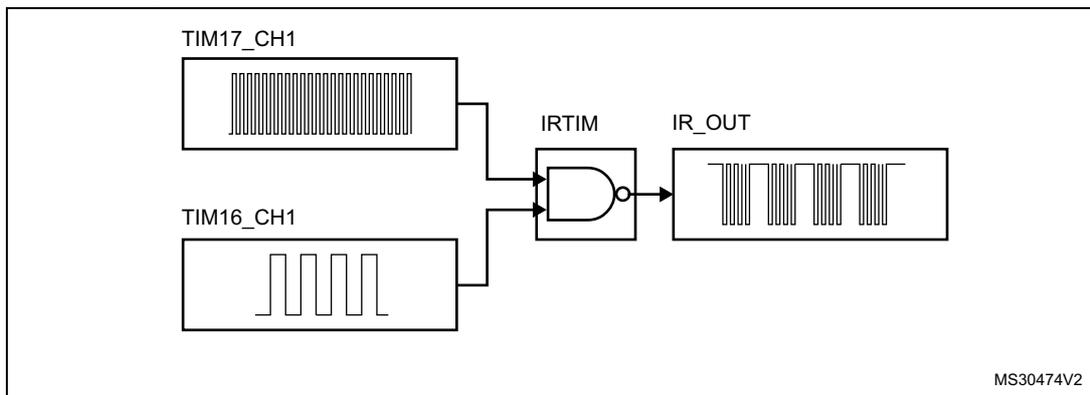
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16 and TIM17 as shown in [Figure 151](#).

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16\_OC1) and TIM17 channel 1 (TIM17\_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

**Figure 151. IR internal hardware connections with TIM16 and TIM17**



All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 generates the modulation envelope.

The infrared function is output on the IR\_OUT pin. The activation of this function is done through the GPIOx\_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 pin) can be activated through the I2C\_PB9\_FMP bit in the SYSCFG\_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

## 20 Basic timers (TIM6/7/18)

### 20.1 Introduction

The basic timers TIM6, TIM7, and TIM18 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.

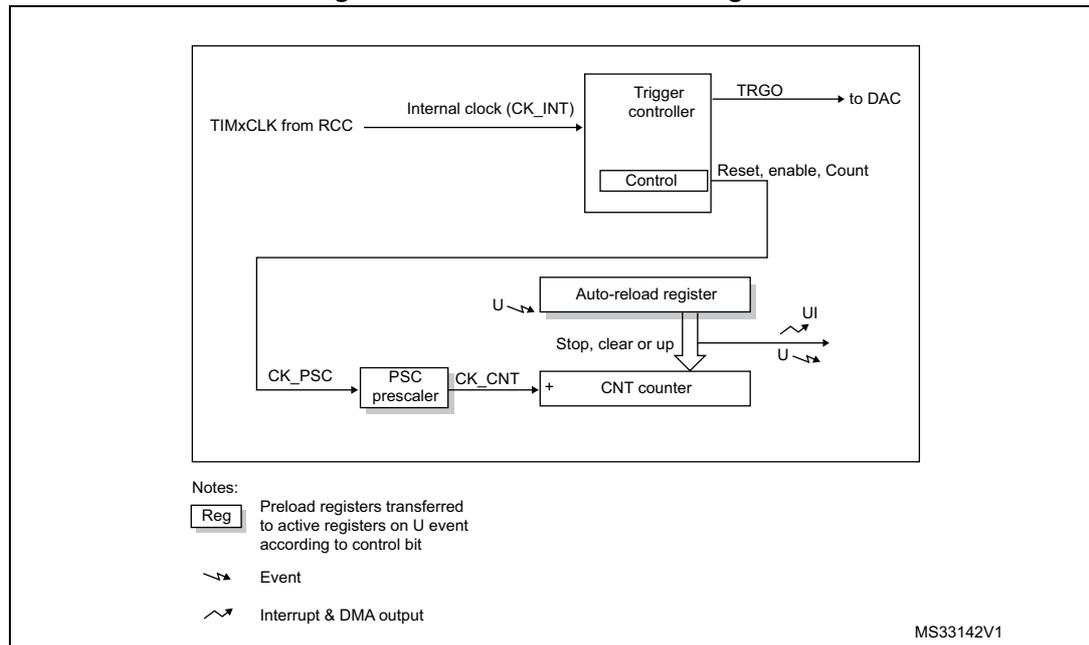
The timers are completely independent, and do not share any resources.

### 20.2 TIM6/7/18 main features

Basic timer (TIM6/TIM7/TIM18) 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 65536
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 152. Basic timer block diagram



## 20.3 TIM6/7/18 functional description

### 20.3.1 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 153](#) and [Figure 154](#) give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 153. Counter timing diagram with prescaler division change from 1 to 2

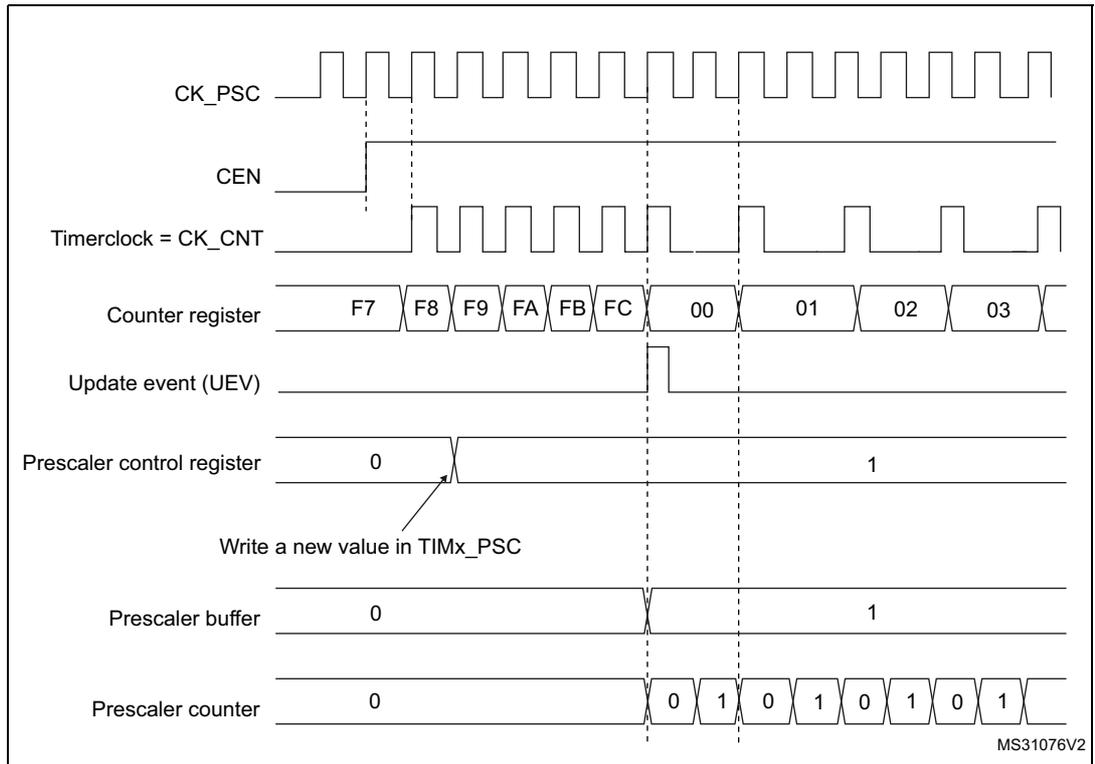
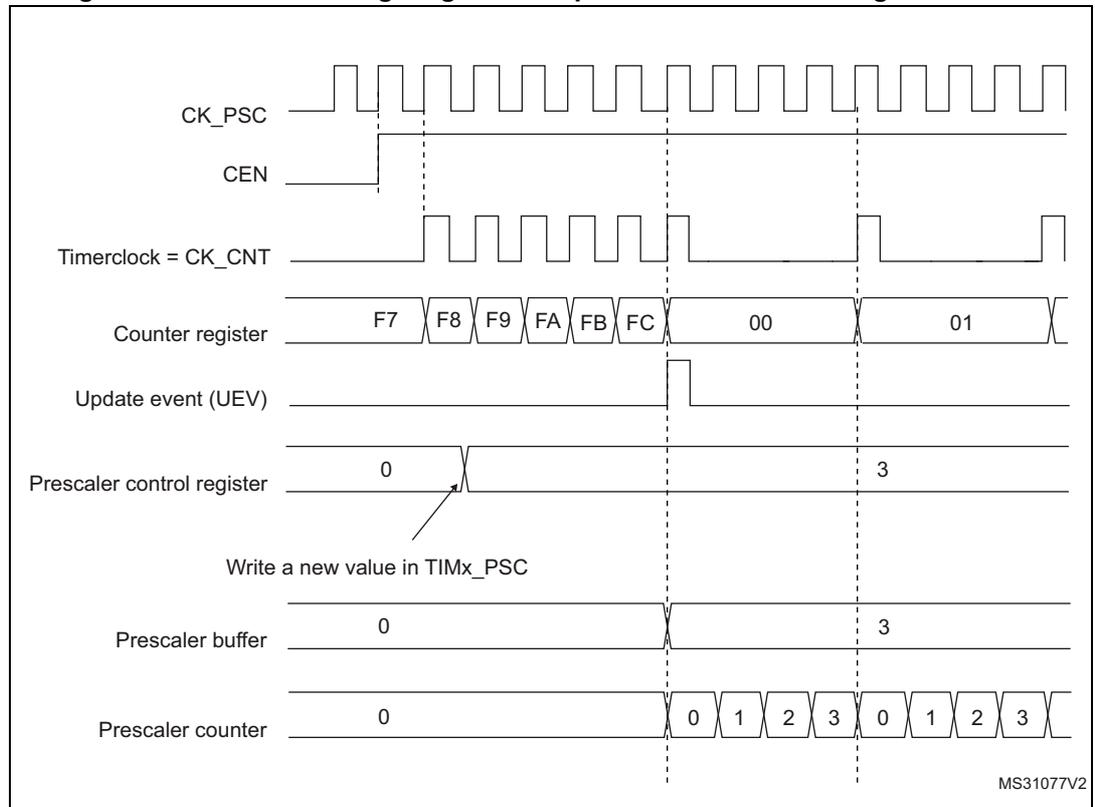


Figure 154. Counter timing diagram with prescaler division change from 1 to 4



### 20.3.2 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 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).

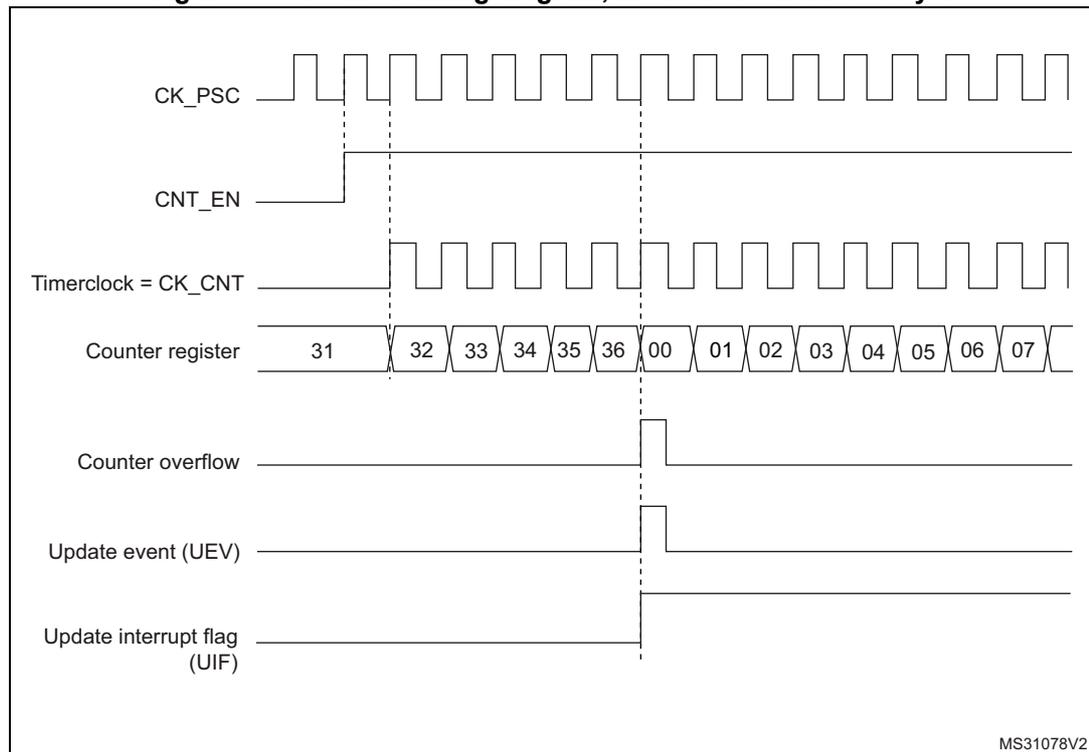
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler 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 155. Counter timing diagram, internal clock divided by 1**



MS31078V2

Figure 156. Counter timing diagram, internal clock divided by 2

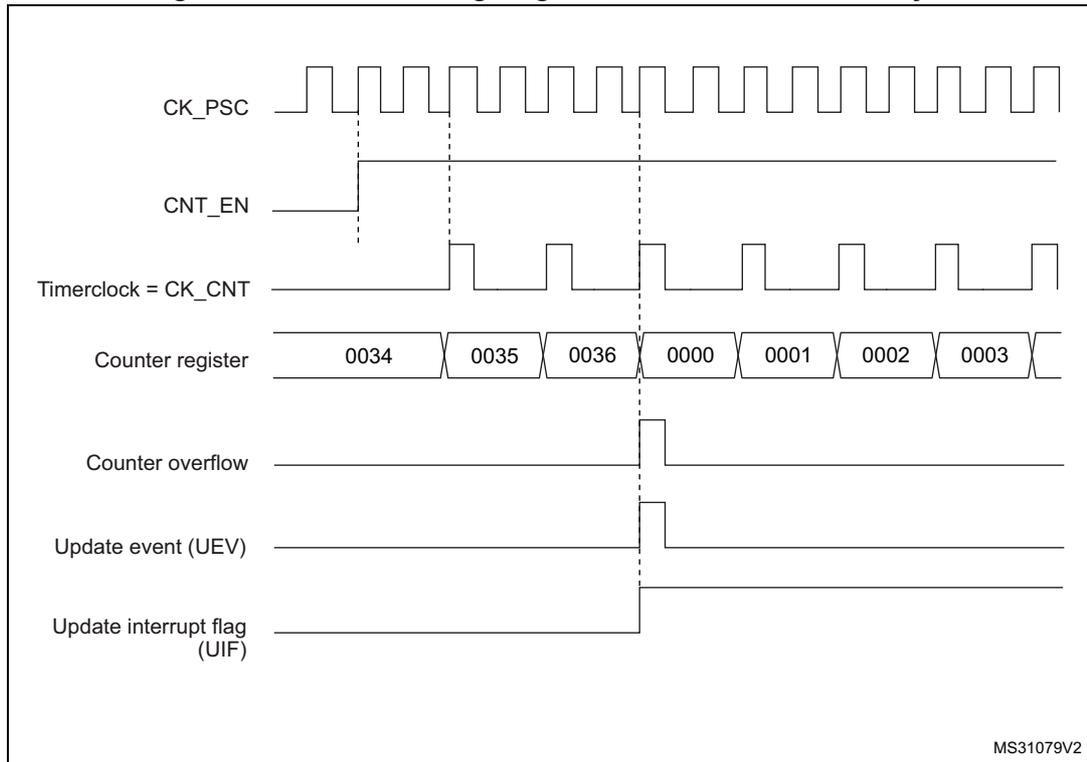


Figure 157. Counter timing diagram, internal clock divided by 4

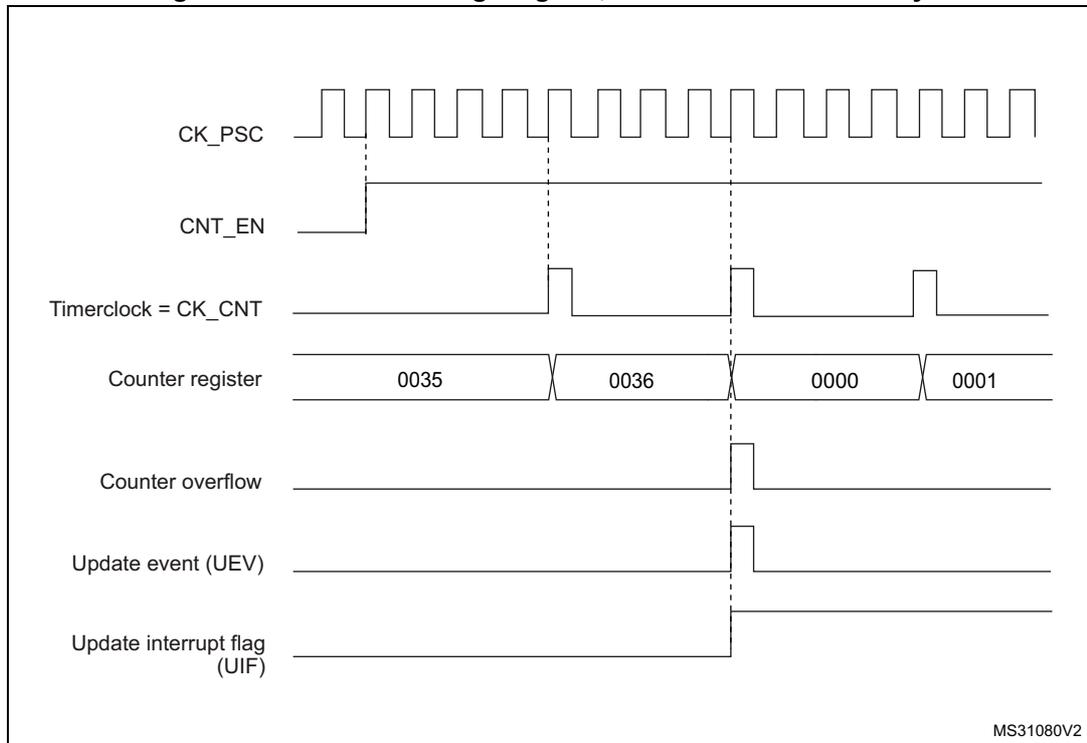
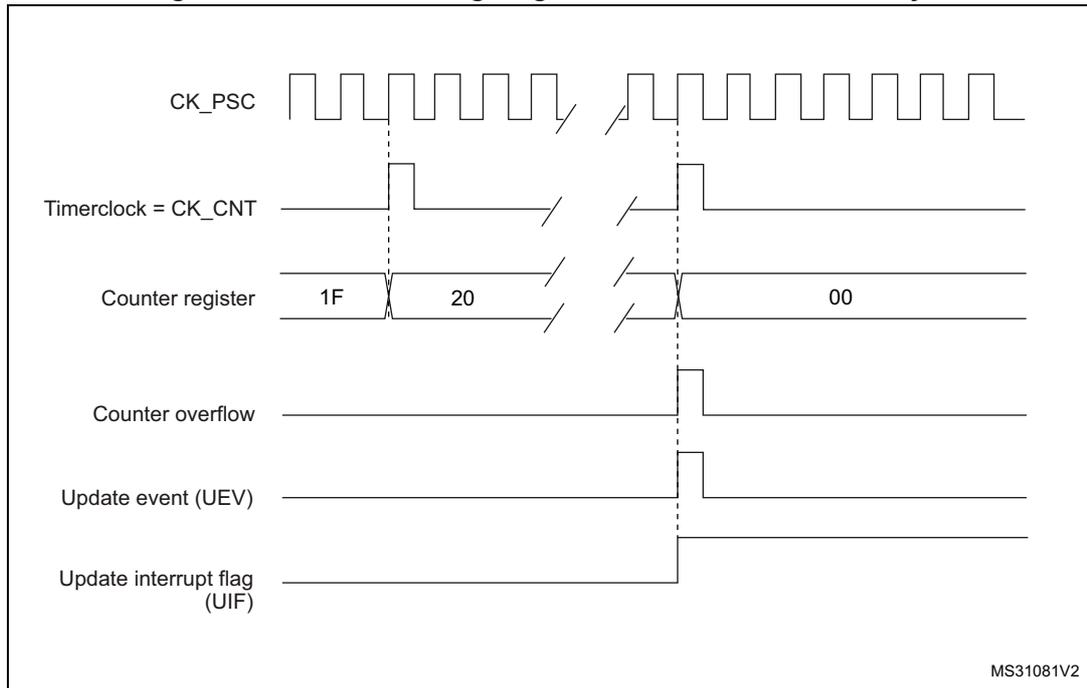
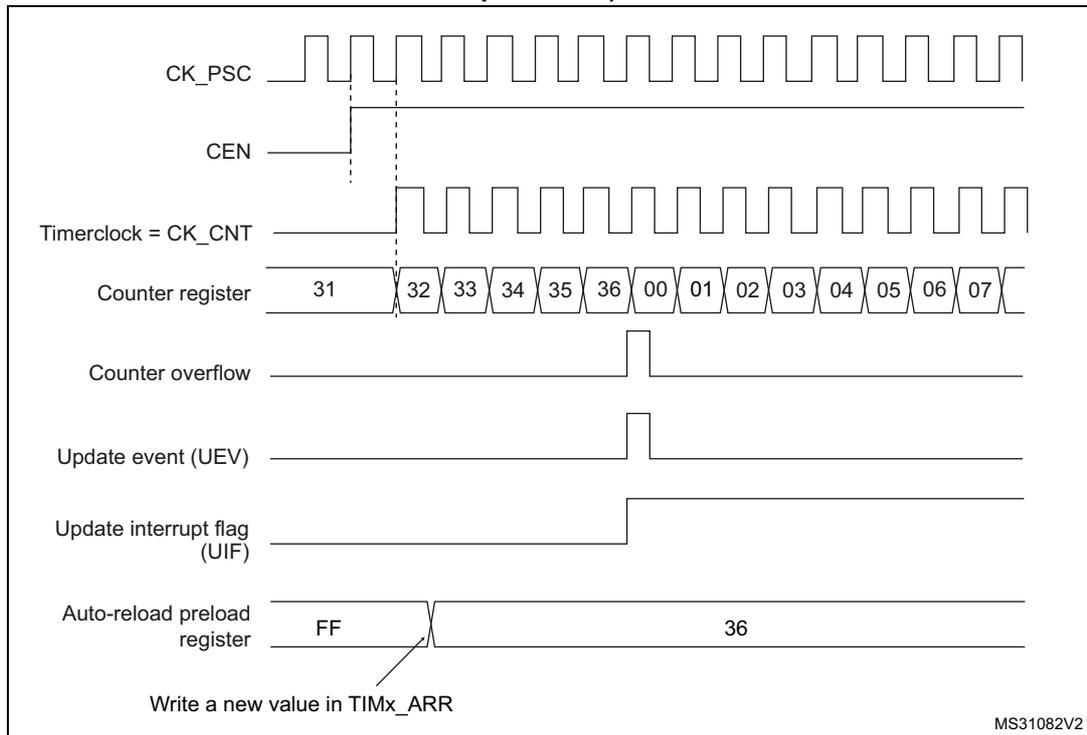


Figure 158. Counter timing diagram, internal clock divided by N



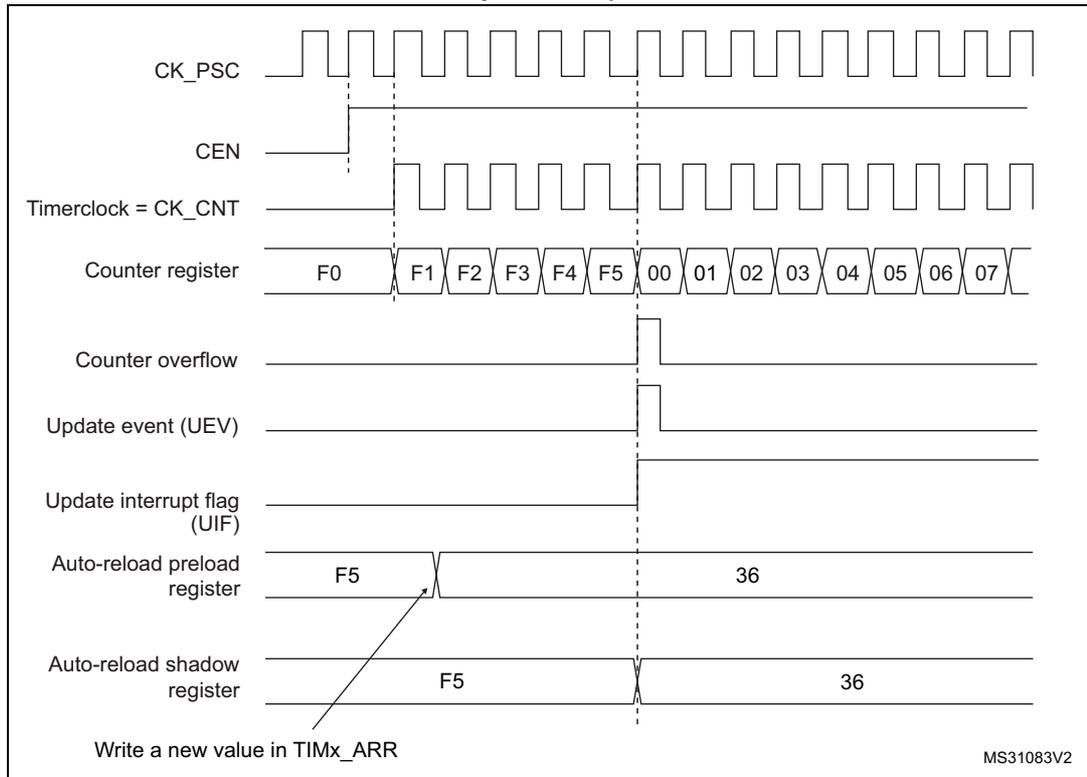
MS31081V2

Figure 159. Counter timing diagram, update event when ARPE = 0 (TIMx\_ARR not preloaded)



MS31082V2

**Figure 160. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



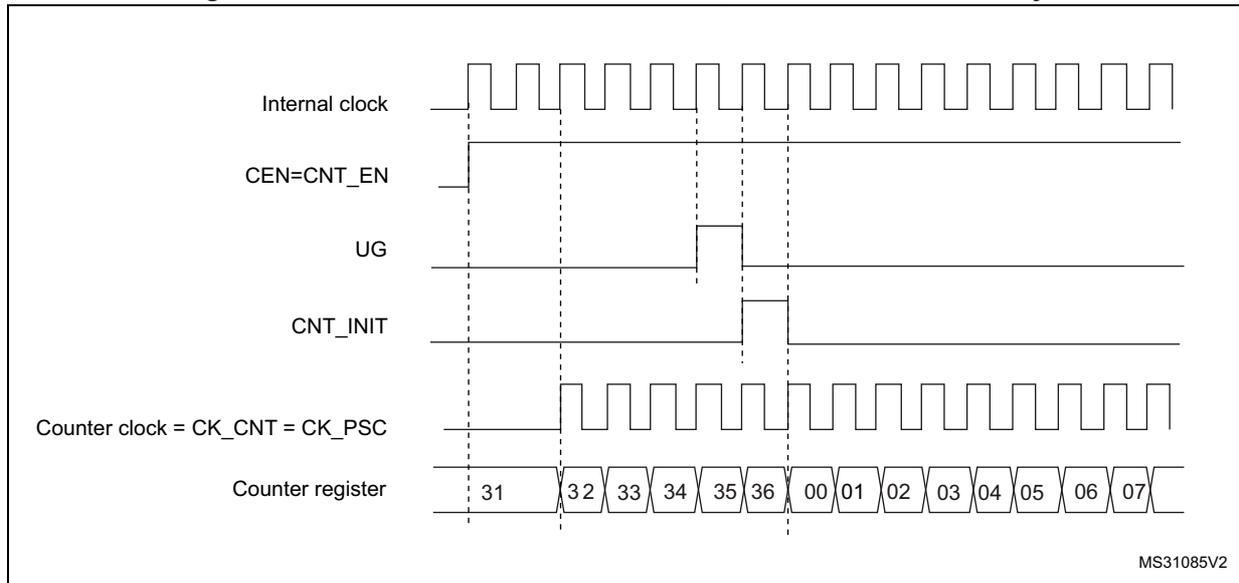
### 20.3.3 Clock source

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 161* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 161. Control circuit in normal mode, internal clock divided by 1



MS31085V2

### 20.3.4 Debug mode

When the microcontroller enters the debug mode (Cortex®-M4 with FPU 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 31.16.2: Debug support for timers, watchdog, bxCAN and I2C](#).

## 20.4 TIM6/7/18 registers

Refer to [Section 1.1: List of abbreviations for registers](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

### 20.4.1 TIM6/7/18 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
Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN							
								rw				rw	rw	rw	rw

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

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx\_ARR register is not buffered.

1: TIMx\_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled.

These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

*Note: Gated mode can work only if the CEN bit has been previously set by software.*

*However trigger mode can set the CEN bit automatically by hardware.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

### 20.4.2 TIM6/7/18 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.	MMS[2:0]			Res.	Res.	Res.	Res.								
									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, must be kept at reset value.

### 20.4.3 TIM6/7/18 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.	UDE	Res.	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.

### 20.4.4 TIM6/7/18 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
Res.	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.

### 20.4.5 TIM6/7/18 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
Res.	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).

### 20.4.6 TIM6/7/18 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **CNT[15:0]**: Counter value

### 20.4.7 TIM6/7/18 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.

### 20.4.8 TIM6/7/18 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 into the actual auto-reload register.

Refer to [Section 20.3.1: Time-base unit on page 480](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 20.4.9 TIM6/7/18 register map

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

**Table 58. TIM6/7/18 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	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN																								
	Reset value																									0				0	0	0	0	
0x04	TIMx_CR2	Res.	MMS[2:0]			Res.	Res.	Res.																										
	Reset value																										0	0	0					
0x08	Res.																																	
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	UJE																										
	Reset value																										0							0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	UIF																										
	Reset value																																	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	UG																										
	Reset value																																	0
0x18	Res.																																	
0x1C	Res.																																	
0x20	Res.																																	
0x24	TIMx_CNT	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]																										
	Reset value																																	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]																										
	Reset value																																	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]																									
	Reset value																																	0

Refer to [Section 2.2.2](#) for the register boundary addresses.



## 21 Independent watchdog (IWDG)

### 21.1 Introduction

The devices feature an embedded watchdog peripheral which offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral serves to detect and resolve malfunctions due to software failure, and to trigger system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to [Section 22 on page 500](#).

### 21.2 IWDG main features

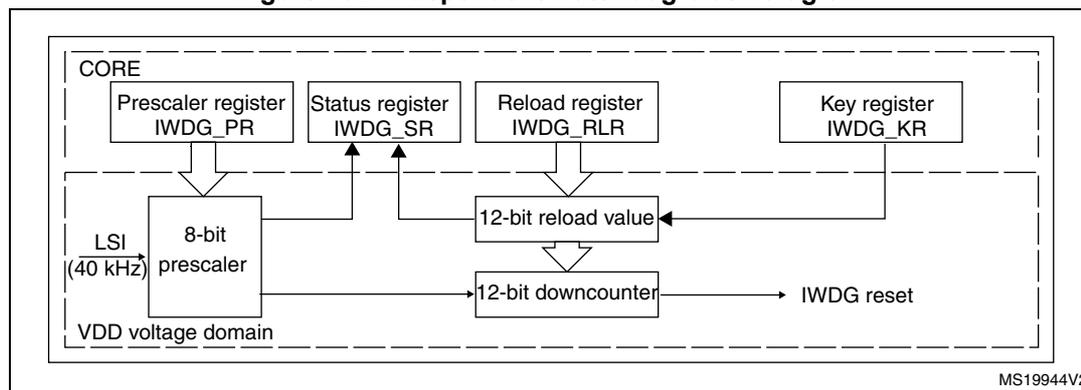
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional Reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 000h
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

### 21.3 IWDG functional description

#### 21.3.1 IWDG block diagram

[Figure 162](#) shows the functional blocks of the independent watchdog module.

**Figure 162. Independent watchdog block diagram**



*Note:* The watchdog function is implemented in the CORE voltage domain that is still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the Key register (IWDG\_KR), the counter starts counting down from the reset value of 0xFFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0x0000 AAAA is written in the IWDG\_KR register, the IWDG\_RLR value is reloaded in the counter and the watchdog reset is prevented.

### 21.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the IWDG\_WINR register.

If the reload operation is performed while the counter is greater than the value stored in the window register (IWDG\_WINR), then a reset is provided.

The default value of the IWDG\_WINR is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the IWDG\_RLR value and ease the cycle number calculation to generate the next reload.

#### Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG\_KR register.
2. Enable register access by writing 0x0000 5555 in the IWDG\_KR register.
3. Write the IWDG prescaler by programming IWDG\_PR from 0 to 7.
4. Write the reload register (IWDG\_RLR).
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Write to the window register IWDG\_WINR. This automatically refreshes the counter value IWDG\_RLR.

*Note:* Writing the window value allows to refresh the Counter value by the RLR when IWDG\_SR is set to 0x0000 0000.

#### Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG\_KR register.
2. Enable register access by writing 0x0000 5555 in the IWDG\_KR register.
3. Write the IWDG prescaler by programming IWDG\_PR from 0 to 7.
4. Write the reload register (IWDG\_RLR).
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Refresh the counter value with IWDG\_RLR (IWDG\_KR = 0x0000 AAAA)

### 21.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the Key register is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

### 21.3.4 Register access protection

Write access to the IWDG\_PR, IWDG\_RLR and IWDG\_WINR registers is protected. To modify them, you must first write the code 0x0000 5555 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 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is on going.

### 21.3.5 Debug mode

When the microcontroller enters debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on DBG\_IWDG\_STOP configuration bit in DBG module.

## 21.4 IWDG registers

Refer to [Section 1.1 on page 36](#) 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 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 15:0 **KEY[15:0]**: Key value (write only, read 0x0000)

These bits must be written by software at regular intervals with the key value 0xAAAA, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 0x5555 to enable access to the IWDG\_PR, IWDG\_RLR and IWDG\_WINR registers (see [Section 21.3.4: Register access protection](#))

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

### 21.4.2 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
Res.	Res.	Res.													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PR[2:0]														
													rw	rw	rw

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

Bits 2:0 **PR[2:0]**: Prescaler divider

These bits are write access protected see [Section 21.3.4: Register access protection](#). 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.*

### 21.4.3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[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 **RL[11:0]**: Watchdog counter reload value

These bits are write access protected see [Section 21.3.4](#). They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG\_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 the datasheet for the timeout information.

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.*

### 21.4.4 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
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WVU	RVU	PVU												
													r	r	r

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

**Bit 2 WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V<sub>DD</sub> voltage domain (takes up to 5 RC 40 kHz cycles).

Window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1

**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<sub>DD</sub> 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<sub>DD</sub> 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, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.*

### 21.4.5 Window register (IWDG\_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF (reset by Standby mode)

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

Bits 31:12 Reserved, must be kept at reset value.

Bits11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected see [Section 21.3.4](#). These bits contain the high limit of the window value to be compared to the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the IWDG\_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 V<sub>DD</sub> voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG\_SR register is reset.*

### 21.4.6 IWDG register map

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

**Table 59. 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	Res	KEY[15:0]																																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	IWDG_PR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR[2:0]																		
	Reset value																																	0	0	0
0x08	IWDG_RLR	Res	Res	Res	Res	Res	RL[11:0]																													
	Reset value																						1	1	1	1	1	1	1	1	1	1	1	1	1	
0x0C	IWDG_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WVU																	
	Reset value																																		0	0
0x10	IWDG_WINR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WIN[11:0]																
	Reset value																							1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 22 System window watchdog (WWDG)

### 22.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications which require the watchdog to react within an accurate timing window.

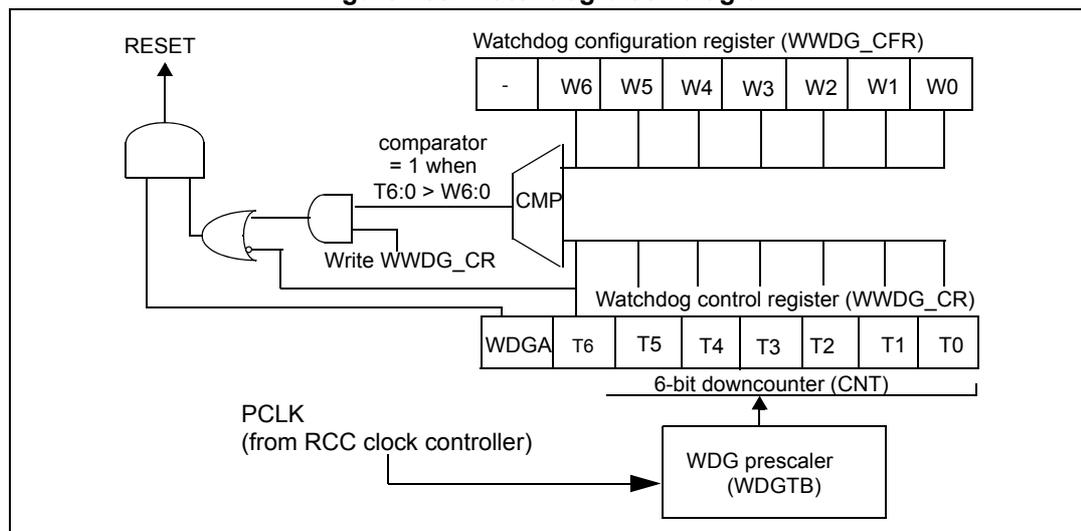
### 22.2 WWDG main features

- 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 164](#))
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

### 22.3 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 163. 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:

### 22.3.1 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.

### 22.3.2 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 164](#)). 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 164](#) 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).

### 22.3.3 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.

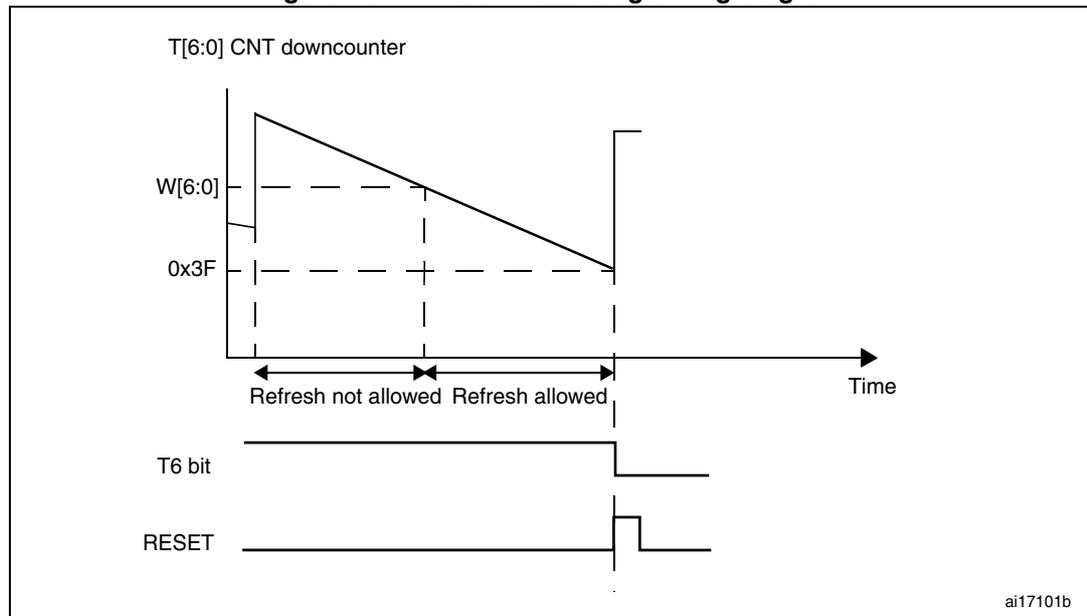
*Note:* When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.

### 22.3.4 How to program the watchdog timeout

You can use the formula in [Figure 164](#) 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 164. Window watchdog timing diagram**



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB} \times (t[5:0] + 1) \quad (\text{ms})$$

where:

$t_{WWDG}$ : WWDG timeout

$t_{PCLK1}$ : APB1 clock period measured in ms

Refer to the datasheet for the minimum and maximum values of the  $T_{WWDG}$ .

### 22.3.5 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M4 with FPU 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 31.16.2: Debug support for timers, watchdog, bxCAN and I2C](#).

## 22.4 WWDG registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 22.4.1 Control register (WWDG\_CR)

Address offset: 0x00

Reset value: 0x0000 007F

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

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

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  $(4096 \times 2^{WDGTB})$  PCLK cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

### 22.4.2 Configuration register (WWDG\_CFR)

Address offset: 0x04

Reset value: 0x0000 007F

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

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

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 (PCLK div 4096) div 1
- 01: CK Counter Clock (PCLK div 4096) div 2
- 10: CK Counter Clock (PCLK div 4096) div 4
- 11: CK Counter Clock (PCLK 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.

### 22.4.3 Status register (WWDG\_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EWIF														
															rc_w0

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

Bit 0 **EWIF**: Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing '0. A write of '1 has no effect. This bit is also set if the interrupt is not enabled.

### 22.4.4 WWDG register map

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

**Table 60. 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	Res.	WDGA	T[6:0]																													
	Reset value																									0	1	1	1	1	1	1	1
0x04	WWDG_CFR	Res.	EWI	WDGTB1	WDGTB0	W[6:0]																											
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF																							
	Reset value																																0

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 23 Real-time clock (RTC)

### 23.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

The RTC includes also a periodic programmable wakeup flag with interrupt capability.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format.

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds, seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After RTC domain reset, all RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

## 23.2 RTC main features

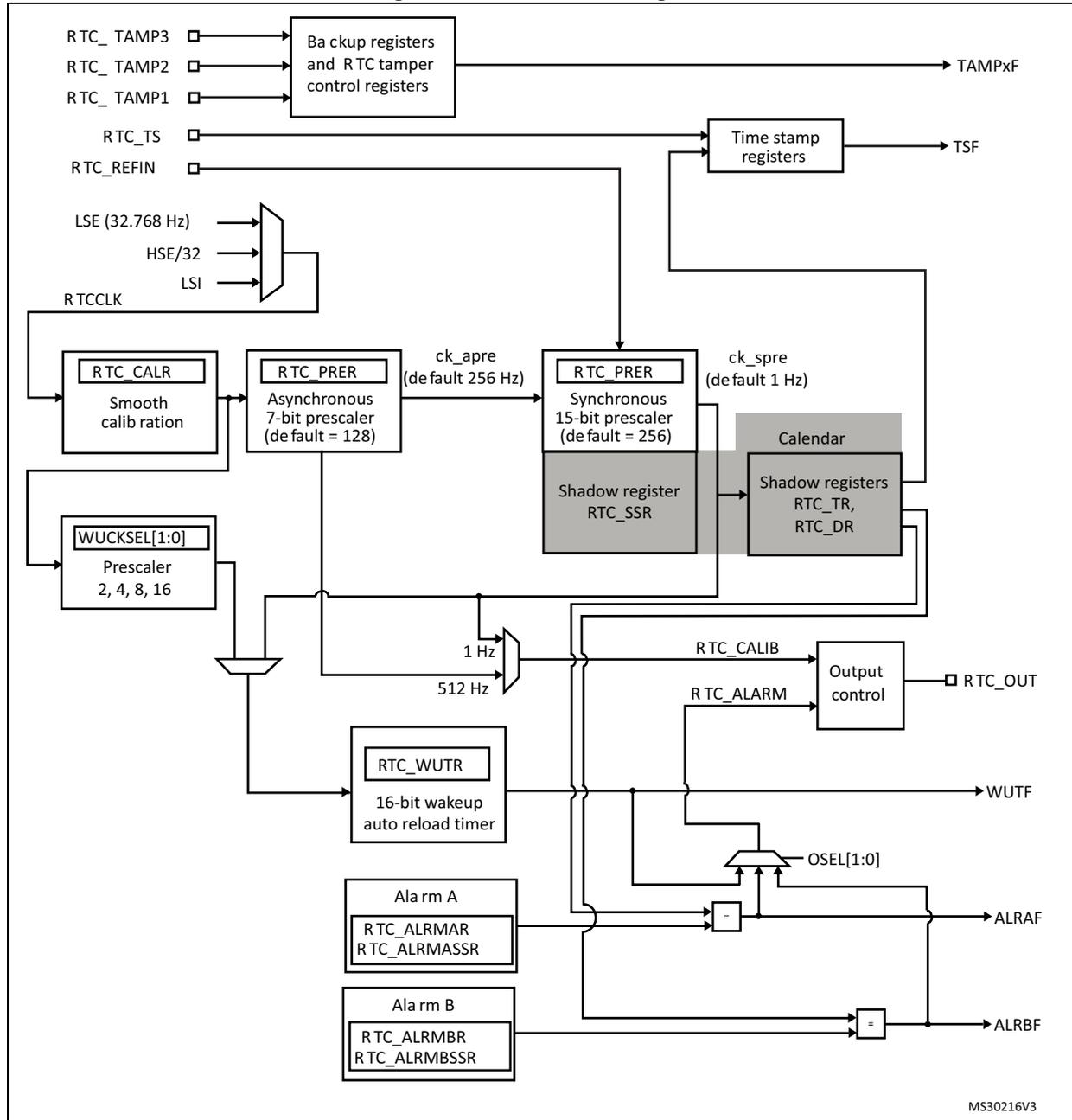
The RTC unit main features are the following (see [Figure 165: RTC block diagram](#)):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Programmable alarm with interrupt function. The alarm can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature.
- Digital calibration circuit (periodic counter correction): 0.95 ppm accuracy, obtained in a calibration window of several seconds
- Time-stamp function for event saving
- Tamper detection event with configurable filter and internal pull-up
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wakeup interrupt
  - Time-stamp
  - Tamper detection
- Backup registers.

### 23.3 RTC functional description

#### 23.3.1 RTC block diagram

Figure 165. RTC block diagram



The RTC includes:

- Two alarms
- Three tamper events
- 32 x 32-bit backup registers
  - The backup registers (RTC\_BKPxR) are implemented in the RTC domain that remains powered-on by VBAT when the VDD power is switched off.
- Alternate function outputs: RTC\_OUT which selects one of the following two outputs:
  - RTC\_CALIB: 512 Hz or 1Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC\_CR register.
  - RTC\_ALARM: This output is enabled by configuring the OSEL[1:0] bits in the RTC\_CR register which select the Alarm A, Alarm B or Wakeup outputs.
- Alternate function inputs:
  - RTC\_TS: timestamp event
  - RTC\_TAMP1: tamper1 event detection
  - RTC\_TAMP2: tamper2 event detection
  - RTC\_TAMP3: tamper3 event detection
  - RTC\_REFIN: 50 or 60 Hz reference clock input

### 23.3.2 GPIOs controlled by the RTC

RTC\_OUT, RTC\_TS and RTC\_TAMP1 are mapped on the same pin (PC13).

The selection of the RTC\_ALARM output is performed through the RTC\_TAFCR register as follows: the PC13VALUE bit is used to select whether the RTC\_ALARM output is configured in push-pull or open drain mode.

When PC13 is not used as RTC alternate function, it can be forced in output push-pull mode by setting the PC13MODE bit in the RTC\_TAFCR. The output data value is then given by the PC13VALUE bit. In this case, PC13 output push-pull state and data are preserved in Standby mode.

The output mechanism follows the priority order shown in [Table 61](#).

When PC14 and PC15 are not used as LSE oscillator, they can be forced in output push-pull mode by setting the PC14MODE and PC15MODE bits in the RTC\_TAFCR register respectively. The output data values are then given by PC14VALUE and PC15VALUE. In this case, the PC14 and PC15 output push-pull states and data values are preserved in Standby mode.

The output mechanism follows the priority order shown in [Table 62](#) and [Table 63](#).

**Table 61. RTC pin PC13 configuration<sup>(1)</sup>**

Pin configuration and function	RTC_ALARM output enabled	RTC_CALIB output enabled	RTC_TAMP1 input enabled	RTC_TS input enabled	PC13MODE bit	PC13VALUE bit
RTC_ALARM output OD	1	Don't care	Don't care	Don't care	Don't care	0
RTC_ALARM output PP	1	Don't care	Don't care	Don't care	Don't care	1

**Table 61. RTC pin PC13 configuration<sup>(1)</sup> (continued)**

Pin configuration and function	RTC_ALARM output enabled	RTC_CALIB output enabled	RTC_TAMP1 input enabled	RTC_TS input enabled	PC13MODE bit	PC13VALUE bit
RTC_CALIB output PP	0	1	Don't care	Don't care	Don't care	Don't care
RTC_TAMP1 input floating	0	0	1	0	Don't care	Don't care
RTC_TS and RTC_TAMP1 input floating	0	0	1	1	Don't care	Don't care
RTC_TS input floating	0	0	0	1	Don't care	Don't care
Output PP forced	0	0	0	0	1	PC13 output data value
Wakeup pin or Standard GPIO	0	0	0	0	0	Don't care

1. OD: open drain; PP: push-pull.

**Table 62. LSE pin PC14 configuration <sup>(1)</sup>**

Pin configuration and function	LSEON bit in RCC_BDCR register	LSEBYP bit in RCC_BDCR register	PC14MODE bit	PC14VALUE bit
LSE oscillator	1	0	Don't care	Don't care
LSE bypass	1	1	Don't care	Don't care
Output PP forced	0	Don't care	1	PC14 output data value
Standard GPIO	0	Don't care	0	Don't care

1. OD: open drain; PP: push-pull.

**Table 63. LSE pin PC15 configuration <sup>(1)</sup>**

Pin configuration and function	LSEON bit in RCC_BDCR register	LSEBYP bit in RCC_BDCR register	PC15MODE bit	PC15VALUE bit
LSE oscillator	1	0	Don't care	Don't care
Output PP forced	1	1	1	PC15 output data value
	0	Don't care		
Standard GPIO	0	Don't care	0	Don't care

1. OD: open drain; PP: push-pull.

### 23.3.3 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock

source configuration, refer to [Section 7: Reset and clock control \(RCC\)](#).

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see [Figure 165: RTC block diagram](#)):

- A 7-bit asynchronous prescaler configured through the PREDIV\_A bits of the RTC\_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV\_S bits of the RTC\_PRER register.

*Note:* When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck\_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is  $2^{22}$ .

This corresponds to a maximum input frequency of around 4 MHz.

$f_{ck\_apre}$  is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck\_apre clock is used to clock the binary RTC\_SSR subseconds downcounter. When it reaches 0, RTC\_SSR is reloaded with the content of PREDIV\_S.

$f_{ck\_spre}$  is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck\_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see [Section 23.3.6: Periodic auto-wakeup](#) for details).

### 23.3.4 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC\_SSR for the subseconds
- RTC\_TR for the time
- RTC\_DR for the date

Every two RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC\_ISR register is set (see [Section 23.6.4: RTC initialization and status register \(RTC\\_ISR\)](#)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the

BYPHAD control bit in the RTC\_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC\_SSR, RTC\_TR or RTC\_DR registers in BYPSHAD=0 mode, the frequency of the APB clock ( $f_{APB}$ ) must be at least 7 times the frequency of the RTC clock ( $f_{RTCCLK}$ ).

The shadow registers are reset by system reset.

### 23.3.5 Programmable alarms

The RTC unit provides programmable alarm: Alarm A and Alarm B. The description below is given for Alarm A, but can be translated in the same way for Alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC\_CR register. The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC\_ALRMASR and RTC\_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC\_ALRMAR register, and through the MASKSSx bits of the RTC\_ALRMASR register. The alarm interrupt is enabled through the ALRAIE bit in the RTC\_CR register.

**Caution:** If the seconds field is selected (MSK0 bit reset in RTC\_ALRMAR), the synchronous prescaler division factor set in the RTC\_PRER register must be at least 3 to ensure correct behavior.

Alarm A and Alarm B (if enabled by bits OSEL[1:0] in RTC\_CR register) can be routed to the RTC\_ALARM output. RTC\_ALARM output polarity can be configured through bit POL the RTC\_CR register.

### 23.3.6 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC\_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.  
When RTCCLK is LSE(32.768kHz), this allows to configure the wakeup interrupt period from 122  $\mu$ s to 32 s, with a resolution down to 61  $\mu$ s.
- ck\_spre (usually 1 Hz internal clock)  
When ck\_spre frequency is 1Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2<sup>16</sup> is added to the 16-bit counter current value. When the initialization sequence is complete (see [Programming the wakeup timer on page 514](#)), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC\_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC\_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC\_CR2 register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the RTC\_ALARM output provided it has been enabled through bits OSEL[1:0] of RTC\_CR register. RTC\_ALARM output polarity can be configured through the POL bit in the RTC\_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

## 23.3.7 RTC initialization and configuration

### RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

### RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by clearing the DBP bit in the PWR\_CR register (refer to the power control section). DBP bit must be set in order to enable RTC registers write access.

After RTC domain reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC\_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC\_TAFCR, RTC\_BKPxR and RTC\_ISR[13:8].

1. Write '0xCA' into the RTC\_WPR register.
2. Write '0x53' into the RTC\_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

### Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC\_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC\_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC\_PRER register.
4. Load the initial time and date values in the shadow registers (RTC\_TR and RTC\_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC\_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

*Note: After a system reset, the application can read the INITS flag in the RTC\_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not*

been initialized since the year field is set at its RTC domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC\_ISR register.

### Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC\_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

### Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for Alarm A but can be translated in the same way for Alarm B.

1. Clear ALRAE in RTC\_CR to disable Alarm A.
2. Program the Alarm A registers (RTC\_ALRMASR/RTC\_ALRMAR).
3. Set ALRAE in the RTC\_CR register to enable Alarm A again.

*Note:* Each change of the RTC\_CR register is taken into account after around 2 RTCCCLK clock cycles due to clock synchronization.

### Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC\_WUTR):

1. Clear WUTE in RTC\_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC\_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC\_CR). Set WUTE in RTC\_CR to enable the timer again. The wakeup timer restarts down-counting.

## 23.3.8 Reading the calendar

### When BYPSHAD control bit is cleared in the RTC\_CR register

To read the RTC calendar registers (RTC\_SSR, RTC\_TR and RTC\_DR) properly, the APB1 clock frequency ( $f_{PCLK}$ ) must be equal to or greater than seven times the RTC clock frequency ( $f_{RTCCCLK}$ ). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC\_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC\_ISR register each time the calendar registers are copied into the RTC\_SSR, RTC\_TR and RTC\_DR shadow registers. The copy is performed every two RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC\_SSR or RTC\_TR locks the values in the higher-order calendar shadow registers until RTC\_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 2 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to [Calendar initialization and configuration on page 513](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After synchronization (refer to [Section 23.3.10: RTC synchronization](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

### When the BYPSHAD control bit is set in the RTC\_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (STOP or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

*Note:* While BYPSHAD=1, instructions which read the calendar registers require one extra APB cycle to complete.

### 23.3.9 Resetting the RTC

The calendar shadow registers (RTC\_SSR, RTC\_TR and RTC\_DR) and some bits of the RTC status register (RTC\_ISR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a RTC domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC\_CR), the prescaler register (RTC\_PRER), the RTC calibration register (RTC\_CALR), the RTC shift register (RTC\_SHIFTR), the RTC timestamp registers (RTC\_TSSSR, RTC\_TSTR and RTC\_TSDR), the RTC tamper and alternate function

configuration register (RTC\_TAFCR), the RTC backup registers (RTC\_BKPxR), the wakeup timer register (RTC\_WUTR), the Alarm A and Alarm B registers (RTC\_ALRMASR/RTC\_ALRMAR and RTC\_ALRMBSSR/RTC\_ALRMBR).

In addition, the RTC keeps on running under system reset if the reset source is different from the RTC domain reset one. When a RTC domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

### 23.3.10 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC\_SSR or RTC\_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC\_SHIFTR.

RTC\_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of  $1 / (\text{PREDIV\_S} + 1)$  seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV\_S[14:0]). The maximum resolution allowed (30.52  $\mu$ s with a 32768 Hz clock) is obtained with PREDIV\_S set to 0x7FFF.

However, increasing PREDIV\_S means that PREDIV\_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC\_SHIFTR). Writing to RTC\_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of  $1 / (\text{PREDIV\_S} + 1)$  seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

**Caution:** Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC\_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

**Caution:** This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC\_SHIFTR when REFCKON=1.

### 23.3.11 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC\_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC\_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC\_REFIN detection is enabled (REFCKON bit of RTC\_CR set to 1), the calendar is still clocked by the LSE, and RTC\_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC\_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts

the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck\_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck\_apre periods when detecting the first reference clock edge. A smaller window of 3 ck\_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck\_apre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck\_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck\_apre period detection window centered on the ck\_spre edge.

When the RTC\_REFIN detection is enabled, PREDIV\_A and PREDIV\_S must be set to their default values:

- PREDIV\_A = 0x007F
- PREDIV\_S = 0x00FF

*Note:* RTC\_REFIN clock detection is not available in Standby mode.

### 23.3.12 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about  $2^{20}$  RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal\_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC\_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

*Note:* CALM[8:0] (RTC\_CALR) specifies the number of RTCCLK pulses to be masked during the 32-second cycle. Setting the bit CALM[0] to '1' causes exactly one pulse to be masked during the 32-second cycle at the moment when cal\_cnt[19:0] is 0x80000; CALM[1]=1 causes two other cycles to be masked (when cal\_cnt is 0x40000 and 0xC0000); CALM[2]=1 causes four other cycles to be masked (cal\_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8]=1 which causes 256 clocks to be masked (cal\_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to '1' effectively inserts an extra RTCCLK pulse every  $2^{11}$  RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency ( $F_{CAL}$ ) given the input frequency ( $F_{RTCCLK}$ ) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [ 1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512) ]$$

### Calibration when PREDIV\_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV\_A bits in RTC\_PRER register) is less than 3. If CALP was already set to 1 and PREDIV\_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV\_A less than 3, the synchronous prescaler value (PREDIV\_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV\_A equals 1 (division factor of 2), PREDIV\_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV\_A equals 0, PREDIV\_S should be set to 32759 rather than 32767 (8 less).

If PREDIV\_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [ 1 + (256 - CALM) / (2^{20} + CALM - 256) ]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

### Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC\_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC\_CALR register can be set to 1 to force a 8- second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

### Re-calibration on-the-fly

The calibration register (RTC\_CALR) can be updated on-the-fly while RTC\_ISR/INITF=0, by using the follow process:

1. Poll the RTC\_ISR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC\_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three  $ck\_apre$  cycles after the write operation to RTC\_CALR, the new calibration settings take effect.

## 23.3.13 Time-stamp function

Time-stamp is enabled by setting the TSE bit of RTC\_CR register to 1.

The calendar is saved in the time-stamp registers (RTC\_TSSSR, RTC\_TSTR, RTC\_TSDR) when a time-stamp event is detected on the RTC\_TS pin.

When a time-stamp event occurs, the time-stamp flag bit (TSF) in RTC\_ISR register is set.

By setting the TSIE bit in the RTC\_CR register, an interrupt is generated when a time-stamp event occurs.

If a new time-stamp event is detected while the time-stamp flag (TSF) is already set, the time-stamp overflow flag (TSOVF) flag is set and the time-stamp registers (RTC\_TSTR and RTC\_TSDR) maintain the results of the previous event.

**Note:** *TSF is set 2  $ck\_apre$  cycles after the time-stamp event occurs due to synchronization process.*

*There is no delay in the setting of TSOVF. This means that if two time-stamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.*

**Caution:** If a time-stamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a time-stamp event occurring at the same moment, the application must not write '0' into TSF bit unless it has already read it to '1'.

Optionally, a tamper event can cause a time-stamp to be recorded. See the description of the TAMPTS control bit in [Section 23.6.14: RTC time-stamp sub second register \(RTC\\_TSSSR\)](#).

### 23.3.14 Tamper detection

The RTC\_TAMPx input events can be configured either for edge detection, or for level detection with filtering.

The tamper detection can be configured for the following purposes:

- erase the RTC backup registers
- generate an interrupt, capable to wakeup from Stop and Standby modes

#### RTC backup registers

The backup registers (RTC\_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs (see [Section 23.6.19: RTC backup registers \(RTC\\_BKPxR\)](#) and [Tamper detection initialization on page 520](#), or when the readout protection of the flash is changed from level 1 to level 0).

#### Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the RTC\_TAFCR register.

Each RTC\_TAMPx tamper detection input is associated with a flag TAMPxF in the RTC\_ISR register.

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck\_apre cycles when TAMPFLT differs from 0x0 (Level detection with filtering)
- 3 ck\_apre cycles when TAMPTS=1 (Timestamp on tamper event)
- No latency when TAMPFLT=0x0 (Edge detection) and TAMPTS=0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

By setting the TAMPIE bit in the RTC\_TAFCR register, an interrupt is generated when a tamper detection event occurs. .

#### Timestamp on tamper event

With TAMPTS set to '1', any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC\_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set at the same time that TSF or TSOVF is set.

#### Edge detection on tamper inputs

If the TAMPFLT bits are "00", the RTC\_TAMPx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the RTC\_TAMPx inputs are deactivated when edge detection is selected.

**Caution:** To avoid losing tamper detection events, the signal used for edge detection is logically ANDed with the corresponding TAMPxE bit in order to detect a tamper detection event in case it occurs before the RTC\_TAMPx pin is enabled.

- When TAMPxTRG = 0: if the RTC\_TAMPx alternate function is already high before tamper detection is enabled (TAMPxE bit set to 1), a tamper event is detected as soon as the RTC\_TAMPx input is enabled, even if there was no rising edge on the RTC\_TAMPx input after TAMPxE was set.
- When TAMPxTRG = 1: if the RTC\_TAMPx alternate function is already low before tamper detection is enabled, a tamper event is detected as soon as the RTC\_TAMPx input is enabled (even if there was no falling edge on the RTC\_TAMPx input after TAMPxE was set).

After a tamper event has been detected and cleared, the RTC\_TAMPx alternate function should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC\_BKPxR). This prevents the application from writing to the backup registers while the RTC\_TAMPx input value still indicates a tamper detection. This is equivalent to a level detection on the RTC\_TAMPx alternate function input.

**Note:** *Tamper detection is still active when  $V_{DD}$  power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the RTC\_TAMPx alternate function is mapped should be externally tied to the correct level.*

### Level detection with filtering on RTC\_TAMPx inputs

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The RTC\_TAMPx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the RTC\_TAMPx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

**Note:** *Refer to the datasheets for the electrical characteristics of the pull-up resistors.*

### 23.3.15 Calibration clock output

When the COE bit is set to 1 in the RTC\_CR register, a reference clock is provided on the RTC\_CALIB device output.

If the COSEL bit in the RTC\_CR register is reset and PREDIV\_A = 0x7F, the RTC\_CALIB frequency is  $f_{RTCCCLK}/64$ . This corresponds to a calibration output at 512 Hz for an RTCCCLK frequency at 32.768 kHz. The RTC\_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and "PREDIV\_S+1" is a non-zero multiple of 256 (i.e: PREDIV\_S[7:0] = 0xFF), the RTC\_CALIB frequency is  $f_{RTCCCLK}/(256 * (PREDIV_A+1))$ . This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV\_A = 0x7F, PREDIV\_S = 0xFF), with an RTCCCLK frequency at 32.768 kHz.

*Note:* When the `RTC_CALIB` or `RTC_ALARM` output is selected, the `RTC_OUT` pin is automatically configured in output alternate function.

### 23.3.16 Alarm output

The `OSEL[1:0]` control bits in the `RTC_CR` register are used to activate the alarm alternate function output `RTC_ALARM`, and to select the function which is output. These functions reflect the contents of the corresponding flags in the `RTC_ISR` register.

The polarity of the output is determined by the `POL` control bit in `RTC_CR` so that the opposite of the selected flag bit is output when `POL` is set to 1.

#### Alarm alternate function output

The `RTC_ALARM` pin can be configured in output open drain or output push-pull using the control bit `ALARMOUTTYPE` in the `RTC_TAFCR` register.

*Note:* Once the `RTC_ALARM` output is enabled, it has priority over `RTC_CALIB` (`COE` bit is don't care and must be kept cleared).

*When the `RTC_CALIB` or `RTC_ALARM` output is selected, the `RTC_OUT` pin is automatically configured in output alternate function.*

## 23.4 RTC low-power modes

**Table 64. Effect of low-power modes on RTC**

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC timestamp event, and RTC Wakeup cause the device to exit the Standby mode.

## 23.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller. Refer to [Section 11.2: Extended interrupts and events controller \(EXTI\)](#).

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Alarm event in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the `RTC_ALARM` IRQ channel in the NVIC.
3. Configure the RTC to generate RTC alarms.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC Tamper event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC\_TAMP\_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the RTC TimeStamp event in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC\_TAMP\_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC time-stamp event.

To enable the Wakeup timer interrupt, the following sequence is required:

1. Configure and enable the EXTI line corresponding to the Wakeup timer even in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the RTC\_WKUP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC Wakeup timer event.

**Table 65. Interrupt control bits**

Interrupt event	Event flag	Enable control bit	Exit from Sleep mode	Exit from Stop mode	Exit from Standby mode
Alarm A	ALRAF	ALRAIE	yes	yes <sup>(1)</sup>	yes <sup>(1)</sup>
Alarm B	ALRBF	ALRBIE	yes	yes <sup>(1)</sup>	yes <sup>(1)</sup>
RTC_TS input (timestamp)	TSF	TSIE	yes	yes <sup>(1)</sup>	yes <sup>(1)</sup>
RTC_TAMP1 input detection	TAMP1F	TAMPIE	yes	yes <sup>(1)</sup>	yes <sup>(1)</sup>
RTC_TAMP2 input detection	TAMP2F	TAMPIE	yes	yes <sup>(1)</sup>	yes <sup>(1)</sup>
RTC_TAMP3 input detection	TAMP3F	TAMPIE	yes	yes <sup>(1)</sup>	yes <sup>(1)</sup>
Wakeup timer interrupt	WUTF	WUTIE	yes	yes <sup>(1)</sup>	yes <sup>(1)</sup>

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.

## 23.6 RTC registers

Refer to [Section 1.1 on page 36](#) of the reference manual for a list of abbreviations used in register descriptions.

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

### 23.6.1 RTC time register (RTC\_TR)

The RTC\_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to [Calendar initialization and configuration on page 513](#) and [Reading the calendar on page 514](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x00

RTC domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31-23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation  
 0: AM or 24-hour format  
 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bit 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bit 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bit 3:0 **SU[3:0]**: Second units in BCD format

### 23.6.2 RTC date register (RTC\_DR)

The RTC\_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to [Calendar initialization and configuration on page 513](#) and [Reading the calendar on page 514](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x04

RTC domain reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
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 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden

001: Monday

...

111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

### 23.6.3 RTC control register (RTC\_CR)

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
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 **COE**: Calibration output enable

This bit enables the RTC\_CALIB output

- 0: Calibration output disabled
- 1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to RTC\_ALARM output

- 00: Output disabled
- 01: Alarm A output enabled
- 10: Alarm B output enabled
- 11: Wakeup output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of RTC\_ALARM output

- 0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])
- 1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection

When COE=1, this bit selects which signal is output on RTC\_CALIB.

- 0: Calibration output is 512 Hz
- 1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV\_A=127 and PREDIV\_S=255). Refer to [Section 23.3.15: Calibration clock output](#)

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

- 0: No effect
- 1: Subtracts 1 hour to the current time. This can be used for winter time change.

- Bit 16 **ADD1H**: Add 1 hour (summer time change)  
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.  
0: No effect  
1: Adds 1 hour to the current time. This can be used for summer time change
- Bit 15 **TSIE**: Time-stamp interrupt enable  
0: Time-stamp Interrupt disable  
1: Time-stamp Interrupt enable
- Bit 14 **WUTIE**: Wakeup timer interrupt enable  
0: Wakeup timer interrupt disabled  
1: Wakeup timer interrupt enabled
- Bit 13 **ALRBIE**: *Alarm B interrupt enable*  
0: Alarm B Interrupt disable  
1: Alarm B Interrupt enable
- Bit 12 **ALRAIE**: Alarm A interrupt enable  
0: Alarm A interrupt disabled  
1: Alarm A interrupt enabled
- Bit 11 **TSE**: timestamp enable  
0: timestamp disable  
1: timestamp enable
- Bit 10 **WUTE**: Wakeup timer enable  
0: Wakeup timer disabled  
1: Wakeup timer enabled
- Bit 9 **ALRBE**: *Alarm B enable*  
0: Alarm B disabled  
1: Alarm B enabled
- Bit 8 **ALRAE**: Alarm A enable  
0: Alarm A disabled  
1: Alarm A enabled
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **FMT**: Hour format  
0: 24 hour/day format  
1: AM/PM hour format
- Bit 5 **BYPHAD**: Bypass the shadow registers  
0: Calendar values (when reading from RTC\_SSR, RTC\_TR, and RTC\_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.  
1: Calendar values (when reading from RTC\_SSR, RTC\_TR, and RTC\_DR) are taken directly from the calendar counters.
- Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to '1'.*

Bit 4 **REFCKON**: RTC\_REFIN reference clock detection enable (50 or 60 Hz)

0: RTC\_REFIN detection disabled

1: RTC\_REFIN detection enabled

*Note: PREDIV\_S must be 0x00FF.*

Bit 3 **TSEDGE**: Time-stamp event active edge

0: RTC\_TS input rising edge generates a time-stamp event

1: RTC\_TS input falling edge generates a time-stamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: Wakeup clock selection

000: RTC/16 clock is selected

001: RTC/8 clock is selected

010: RTC/4 clock is selected

011: RTC/2 clock is selected

10x: ck\_spre (usually 1 Hz) clock is selected

11x: ck\_spre (usually 1 Hz) clock is selected and  $2^{16}$  is added to the WUT counter value (see note below)

*Note: Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC\_ISR/INITF = 1).*

*WUT = Wakeup unit counter value.  $WUT = (0x0000 \text{ to } 0xFFFF) + 0x10000$  added when  $WUCKSEL[2:1 = 11]$ .*

*Bits 2 to 0 of this register can be written only when RTC\_CR WUTE bit = 0 and RTC\_ISR WUTWF bit = 1.*

*It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.*

*ADD1H and SUB1H changes are effective in the next second.*

*This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).*

### 23.6.4 RTC initialization and status register (RTC\_ISR)

This register is write protected (except for RTC\_ISR[13:8] bits). The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x0C

RTC domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to '0'

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

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

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to '1' when software writes to the RTC\_CALR register, indicating that the RTC\_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to '0'. Refer to [Re-calibration on-the-fly](#).

Bit 15 **TAMP3F**: RTC\_TAMP3 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC\_TAMP3 input.

It is cleared by software writing 0

Bit 14 **TAMP2F**: RTC\_TAMP2 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC\_TAMP2 input.

It is cleared by software writing 0

Bit 13 **TAMP1F**: RTC\_TAMP1 detection flag

This flag is set by hardware when a tamper detection event is detected on the RTC\_TAMP1 input.

It is cleared by software writing 0

Bit 12 **TSOVF**: Time-stamp overflow flag

This flag is set by hardware when a time-stamp event occurs while TSF is already set.

This flag is cleared by software by writing 0. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a time-stamp event occurs immediately before the TSF bit is cleared.

Bit 11 **TSF**: Time-stamp flag

This flag is set by hardware when a time-stamp event occurs.

This flag is cleared by software by writing 0.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.

This flag is cleared by software by writing 0.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

- Bit 9 **ALRBF**: Alarm B flag  
This flag is set by hardware when the time/date registers (RTC\_TR and RTC\_DR) match the Alarm B register (RTC\_ALRMBR).  
This flag is cleared by software by writing 0.
- Bit 8 **ALRAF**: Alarm A flag  
This flag is set by hardware when the time/date registers (RTC\_TR and RTC\_DR) match the Alarm A register (RTC\_ALRMAR).  
This flag is cleared by software by writing 0.
- Bit 7 **INIT**: Initialization mode  
0: Free running mode  
1: Initialization mode used to program time and date register (RTC\_TR and RTC\_DR), and prescaler register (RTC\_PRER). Counters are stopped and start counting from the new value when INIT is reset.
- Bit 6 **INITF**: Initialization flag  
When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.  
0: Calendar registers update is not allowed  
1: Calendar registers update is allowed
- Bit 5 **RSF**: Registers synchronization flag  
This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC\_SSRx, RTC\_TRx and RTC\_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.  
It is cleared either by software or by hardware in initialization mode.  
0: Calendar shadow registers not yet synchronized  
1: Calendar shadow registers synchronized
- Bit 4 **INITS**: Initialization status flag  
This bit is set by hardware when the calendar year field is different from 0 (RTC domain reset state).  
0: Calendar has not been initialized  
1: Calendar has been initialized
- Bit 3 **SHPF**: Shift operation pending  
0: No shift operation is pending  
1: A shift operation is pending  
This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC\_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

**Bit 2 WUTWF:** Wakeup timer write flag

This bit is set by hardware when the wakeup timer values can be changed, after the WUTE bit has been set to 0 in RTC\_CR.

- 0: Wakeup timer configuration update not allowed
- 1: Wakeup timer configuration update allowed

**Bit 1 ALRBWF:** Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC\_CR.

It is cleared by hardware in initialization mode.

- 0: Alarm B update not allowed
- 1: Alarm B update allowed

**Bit 0 ALRAWF:** Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC\_CR.

It is cleared by hardware in initialization mode.

- 0: Alarm A update not allowed
- 1: Alarm A update allowed

*Note:* The bits ALRAF, ALRBF, WUTF and TSF are cleared 2 APB clock cycles after programming them to 0.

### 23.6.5 RTC prescaler register (RTC\_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to [Calendar initialization and configuration on page 513](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x10

RTC domain reset value: 0x007F 00FF

System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value

Bits 22:16 **PREDIV\_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:

$$ck\_apre\ frequency = RTCCLK\ frequency / (PREDIV\_A + 1)$$

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV\_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:

$$ck\_spre\ frequency = ck\_apre\ frequency / (PREDIV\_S + 1)$$

### 23.6.6 RTC wakeup timer register (RTC\_WUTR)

This register can be written only when WUTWF is set to 1 in RTC\_ISR.

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x14

RTC domain reset value: 0x0000 FFFF

System reset: not affected

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

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

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck\_wut cycles. The ck\_wut period is selected through WUCKSEL[2:0] bits of the RTC\_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs (WUT+1) ck\_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

### 23.6.7 RTC alarm A register (RTC\_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC\_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x1C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
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
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask  
 0: Alarm A set if the date/day match  
 1: Date/day don't care in Alarm A comparison

Bit 30 **WDSEL**: Week day selection  
 0: DU[3:0] represents the date units  
 1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format.

Bits 27:24 **DU[3:0]**: Date units or day in BCD format.

Bit 23 **MSK3**: Alarm A hours mask  
 0: Alarm A set if the hours match  
 1: Hours don't care in Alarm A comparison

Bit 22 **PM**: AM/PM notation  
 0: AM or 24-hour format  
 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 **MSK2**: Alarm A minutes mask  
 0: Alarm A set if the minutes match  
 1: Minutes don't care in Alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 **MSK1**: Alarm A seconds mask  
 0: Alarm A set if the seconds match  
 1: Seconds don't care in Alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

### 23.6.8 RTC alarm B register (RTC\_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC\_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x20

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
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
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask  
 0: Alarm B set if the date and day match  
 1: Date and day don't care in Alarm B comparison

Bit 30 **WDSEL**: Week day selection  
 0: DU[3:0] represents the date units  
 1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask  
 0: Alarm B set if the hours match  
 1: Hours don't care in Alarm B comparison

Bit 22 **PM**: AM/PM notation  
 0: AM or 24-hour format  
 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask  
 0: Alarm B set if the minutes match  
 1: Minutes don't care in Alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask  
 0: Alarm B set if the seconds match  
 1: Seconds don't care in Alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

### 23.6.9 RTC write protection register (RTC\_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 7:0 **KEY**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to [RTC register write protection](#) for a description of how to unlock RTC register write protection.

**23.6.10 RTC sub second register (RTC\_SSR)**

Address offset: 0x28

RTC domain reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

Bits31:16 Reserved, must be kept at reset value

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

$$\text{Second fraction} = (\text{PREDIV\_S} - \text{SS}) / (\text{PREDIV\_S} + 1)$$

*Note: SS can be larger than PREDIV\_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC\_TR/RTC\_DR.*

### 23.6.11 RTC shift control register (RTC\_SHIFTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC\_ISR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value

Bits 14:0 **SUBFS**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF=1, in RTC\_ISR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

$$\text{Delay (seconds)} = \text{SUBFS} / (\text{PREDIV}_S + 1)$$

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

$$\text{Advance (seconds)} = (1 - (\text{SUBFS} / (\text{PREDIV}_S + 1)))$$

*Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.*

### 23.6.12 RTC timestamp time register (RTC\_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC\_ISR. It is cleared when TSF bit is reset.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format  
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

### 23.6.13 RTC timestamp date register (RTC\_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC\_ISR. It is cleared when TSF bit is reset.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

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

Bits 15:13 **WDU[1:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bit 3:0 **DU[3:0]**: Date units in BCD format

**23.6.14 RTC time-stamp sub second register (RTC\_TSSSR)**

The content of this register is valid only when RTC\_ISR/TSF is set. It is cleared when the RTC\_ISR/TSF bit is reset.

Address offset: 0x38

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[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 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

### 23.6.15 RTC calibration register (RTC\_CALR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#).

Address offset: 0x3C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:16 Reserved, must be kept at reset value

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every  $2^{11}$  pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:  $(512 * CALP) - CALM$ .

Refer to [Section 23.3.12: RTC smooth digital calibration](#).

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to '1', the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at "00" when CALW8='1'. Refer to [Section 23.3.12: RTC smooth digital calibration](#).

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to '1', the 16-second calibration cycle period is selected. This bit must not be set to '1' if CALW8=1.

Note: CALM[0] is stuck at '0' when CALW16='1'. Refer to [Section 23.3.12: RTC smooth digital calibration](#).

Bits 12:9 Reserved, must be kept at reset value

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of  $2^{20}$  RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See [Section 23.3.12: RTC smooth digital calibration on page 517](#).

### 23.6.16 RTC tamper and alternate function configuration register (RTC\_TAFCR)

Address offset: 0x40

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC15 MODE	PC15 VALUE	PC14 MODE	PC14 VALUE	PC13 MODE	PC13 VALUE	Res.	Res.
								rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMPP UDIS	TAMPPRCH [1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPT S	TAMP3 -TRG	TAMP3 E	TAMP2 TRG	TAMP2 E	TAMPIE	TAMP1 TRG	TAMP1 E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **PC15MODE**: PC15 mode

0: PC15 is controlled by the GPIO configuration registers. Consequently PC15 is floating in Standby mode.

1: PC15 is forced to push-pull output if LSE is disabled.

Bit 22 **PC15VALUE**: PC15 value

If the LSE is disabled and PC15MODE = 1, PC15VALUE configures the PC15 output data.

Bit 21 **PC14MODE**: PC14 mode

0: PC14 is controlled by the GPIO configuration registers. Consequently PC14 is floating in Standby mode.

1: PC14 is forced to push-pull output if LSE is disabled.

Bit 20 **PC14VALUE**: PC14 value

If the LSE is disabled and PC14MODE = 1, PC14VALUE configures the PC14 output data.

Bit 19 **PC13MODE**: PC13 mode

0: PC13 is controlled by the GPIO configuration registers. Consequently PC13 is floating in Standby mode.

1: PC13 is forced to push-pull output if all RTC alternate functions are disabled.

Bit 18 **PC13VALUE**: RTC\_ALARM output type/PC13 value

If PC13 is used to output RTC\_ALARM, PC13VALUE configures the output configuration:

0: RTC\_ALARM is an open-drain output

1: RTC\_ALARM is a push-pull output

If all RTC alternate functions are disabled and PC13MODE = 1, PC13VALUE configures the PC13 output data.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **TAMPPUDIS**: RTC\_TAMPx pull-up disable

This bit determines if each of the RTC\_TAMPx pins are pre-charged before each sample.

0: Precharge RTC\_TAMPx pins before sampling (enable internal pull-up)

1: Disable precharge of RTC\_TAMPx pins.

- Bits 14:13 **TAMPPRCH[1:0]**: RTC\_TAMPx precharge duration  
 These bits determine the duration of time during which the pull-up is activated before each sample. TAMPPRCH is valid for each of the RTC\_TAMPx inputs.  
 0x0: 1 RTCCLK cycle  
 0x1: 2 RTCCLK cycles  
 0x2: 4 RTCCLK cycles  
 0x3: 8 RTCCLK cycles
- Bits 12:11 **TAMPFLT[1:0]**: RTC\_TAMPx filter count  
 These bits determine the number of consecutive samples at the specified level (TAMP\*TRG) needed to activate a Tamper event. TAMPFLT is valid for each of the RTC\_TAMPx inputs.  
 0x0: Tamper event is activated on edge of RTC\_TAMPx input transitions to the active level (no internal pull-up on RTC\_TAMPx input).  
 0x1: Tamper event is activated after 2 consecutive samples at the active level.  
 0x2: Tamper event is activated after 4 consecutive samples at the active level.  
 0x3: Tamper event is activated after 8 consecutive samples at the active level.
- Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency  
 Determines the frequency at which each of the RTC\_TAMPx inputs are sampled.  
 0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)  
 0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)  
 0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)  
 0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)  
 0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)  
 0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)  
 0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)  
 0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)
- Bit 7 **TAMPTS**: Activate timestamp on tamper detection event  
 0: Tamper detection event does not cause a timestamp to be saved  
 1: Save timestamp on tamper detection event  
 TAMPTS is valid even if TSE=0 in the RTC\_CR register.
- Bit 6 **TAMP3TRG**: Active level for RTC\_TAMP3 input  
 if TAMPFLT != 00:  
 0: RTC\_TAMP3 input staying low triggers a tamper detection event.  
 1: RTC\_TAMP3 input staying high triggers a tamper detection event.  
 if TAMPFLT = 00:  
 0: RTC\_TAMP3 input rising edge triggers a tamper detection event.  
 1: RTC\_TAMP3 input falling edge triggers a tamper detection event.
- Bit 5 **TAMP3E**: RTC\_TAMP3 detection enable  
 0: RTC\_TAMP3 input detection disabled  
 1: RTC\_TAMP3 input detection enabled
- Bit 4 **TAMP2TRG**: Active level for RTC\_TAMP2 input  
 if TAMPFLT != 00:  
 0: RTC\_TAMP2 input staying low triggers a tamper detection event.  
 1: RTC\_TAMP2 input staying high triggers a tamper detection event.  
 if TAMPFLT = 00:  
 0: RTC\_TAMP2 input rising edge triggers a tamper detection event.  
 1: RTC\_TAMP2 input falling edge triggers a tamper detection event.

- Bit 3 **TAMP2E**: RTC\_TAMP2 input detection enable  
0: RTC\_TAMP2 detection disabled  
1: RTC\_TAMP2 detection enabled
- Bit 2 **TAMPIE**: Tamper interrupt enable  
0: Tamper interrupt disabled  
1: Tamper interrupt enabled.
- Bit 1 **TAMP1TRG**: Active level for RTC\_TAMP1 input  
If TAMPFLT != 00  
0: RTC\_TAMP1 input staying low triggers a tamper detection event.  
1: RTC\_TAMP1 input staying high triggers a tamper detection event.  
if TAMPFLT = 00:  
0: RTC\_TAMP1 input rising edge triggers a tamper detection event.  
1: RTC\_TAMP1 input falling edge triggers a tamper detection event.
- Bit 0 **TAMP1E**: RTC\_TAMP1 input detection enable  
0: RTC\_TAMP1 detection disabled  
1: RTC\_TAMP1 detection enabled

**Caution:** When TAMPFLT = 0, TAMP1E must be reset when TAMP1TRG is changed to avoid spuriously setting TAMP1F.

### 23.6.17 RTC alarm A sub second register (RTC\_ALRMASR)

This register can be written only when ALRAE is reset in RTC\_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 513](#)

Address offset: 0x44

RTC domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 27:24 MASKSS[3:0]: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.

...

12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.

13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.

14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.

15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits23:15 Reserved, must be kept at reset value.

Bits 14:0 SS[14:0]: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

### 23.6.18 RTC alarm B sub second register (RTC\_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC\_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

Address offset: 0x48

RTC domain reset value: 0x0000 0000

System reset: not affected

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

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

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

0x1: SS[14:1] are don't care in Alarm B comparison. Only SS[0] is compared.

0x2: SS[14:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

0x3: SS[14:3] are don't care in Alarm B comparison. Only SS[2:0] are compared.

...

0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared.

0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared.

0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared.

0xF: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

### 23.6.19 RTC backup registers (RTC\_BKPxR)

Address offset: 0x50 to 0xCC

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers.

They are powered-on by V<sub>BAT</sub> when V<sub>DD</sub> is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.

This register is reset on a tamper detection event, as long as TAMPx<sub>F</sub>=1. or when the Flash readout protection is disabled.

23.6.20 RTC register map

Table 66. 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_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]	HU[3:0]			Res.	MNT[2:0]			MNU[3:0]			Res.	ST[2:0]			SU[3:0]							
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]			YU[3:0]			WDU[2:0]		MT	MU[3:0]			Res.	Res.	DT [1:0]	DU[3:0]								
	Reset value										0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1		0	0	0	0	0	1	
0x08	RTC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COE	OSE [1:0]	POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPHAD	REFCKON	TSEDGE	WUCKSEL[2:0]			
	Reset value									0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	RTC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTF	ALBWF	ALRAWF	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						PREDIV_S[14:0]																	
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
0x14	RTC_WUTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]																
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]	DU[3:0]			MSK3	PM	HT [1:0]	HU[3:0]			MSK2	MNT[2:0]		MNU[3:0]			MSK1	ST[2:0]		SU[3:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]	DU[3:0]			MSK3	PM	HT [1:0]	HU[3:0]			MSK2	MNT[2:0]		MNU[3:0]			MSK2	ST[2:0]		SU[3:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY								
	Reset value																									0	0	0	0	0	0	0	0	0
0x28	RTC_SSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	SUBFS[14:0]																															
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]	HU[3:0]			Res.	MNT[2:0]		MNU[3:0]			Res.	ST[2:0]		SU[3:0]									
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]	MT	MU[3:0]			Res.	Res.	DT [1:0]	DU[3:0]								
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x38	RTC_TSSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 66. RTC 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					
0x3C	RTC_CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]													
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0	0				
0x40	RTC_TAFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC15MODE	PC15MODE	PC14VALUE	PC14MODE	PC13VALUE	PC13VALUE	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFL[1:0]		TAMPFREQ[2:0]		TAMPTS	TAMP3-TRG	TAMP3E	TAMP2-TRG	TAMP2E	TAMP1E	TAMP1TRG	TAMP1E						
	Reset value									0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x44	RTC_ALRMASR	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]																			
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x48	RTC_ALRMASR	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]																			
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x50 to 0xCC	RTC_BKP0R	BKP[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					
	to RTC_BKP31R	BKP[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					

Refer to [Section 2.2.2](#) for the register boundary addresses.



## 24 Inter-integrated circuit (I2C) interface

### 24.1 Introduction

The I<sup>2</sup>C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I<sup>2</sup>C bus. It provides multimaster capability, and controls all I<sup>2</sup>C bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

It is also SMBus (system management bus) and PMBus (power management bus) compatible.

DMA can be used to reduce CPU overload.

### 24.2 I2C main features

- I<sup>2</sup>C bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy to use event management
  - Optional clock stretching
  - Software reset
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters

The following additional features are also available depending on the product implementation (see [Section 24.3: I2C implementation](#)):

- SMBus specification rev 2.0 compatibility:
  - Hardware PEC (Packet Error Checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and Device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.1 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I2C communication speed to be independent from the PCLK reprogramming
- Wakeup from Stop mode on address match.

## 24.3 I2C implementation

This manual describes the full set of features implemented in I2C1. In the STM32F37xx devices both I2C1 and I2C2 are identical and implement the full set of features as shown in the following table.

**Table 67. STM32F37xx I2C implementation**

I2C features <sup>(1)</sup>	I2C1	I2C2
Independent clock	X	X
SMBus	X	X
Wakeup from Stop mode	X	X
20 mA output drive for Fm+ mode	X	X

1. X = supported.

## 24.4 I2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I<sup>2</sup>C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I<sup>2</sup>C bus.

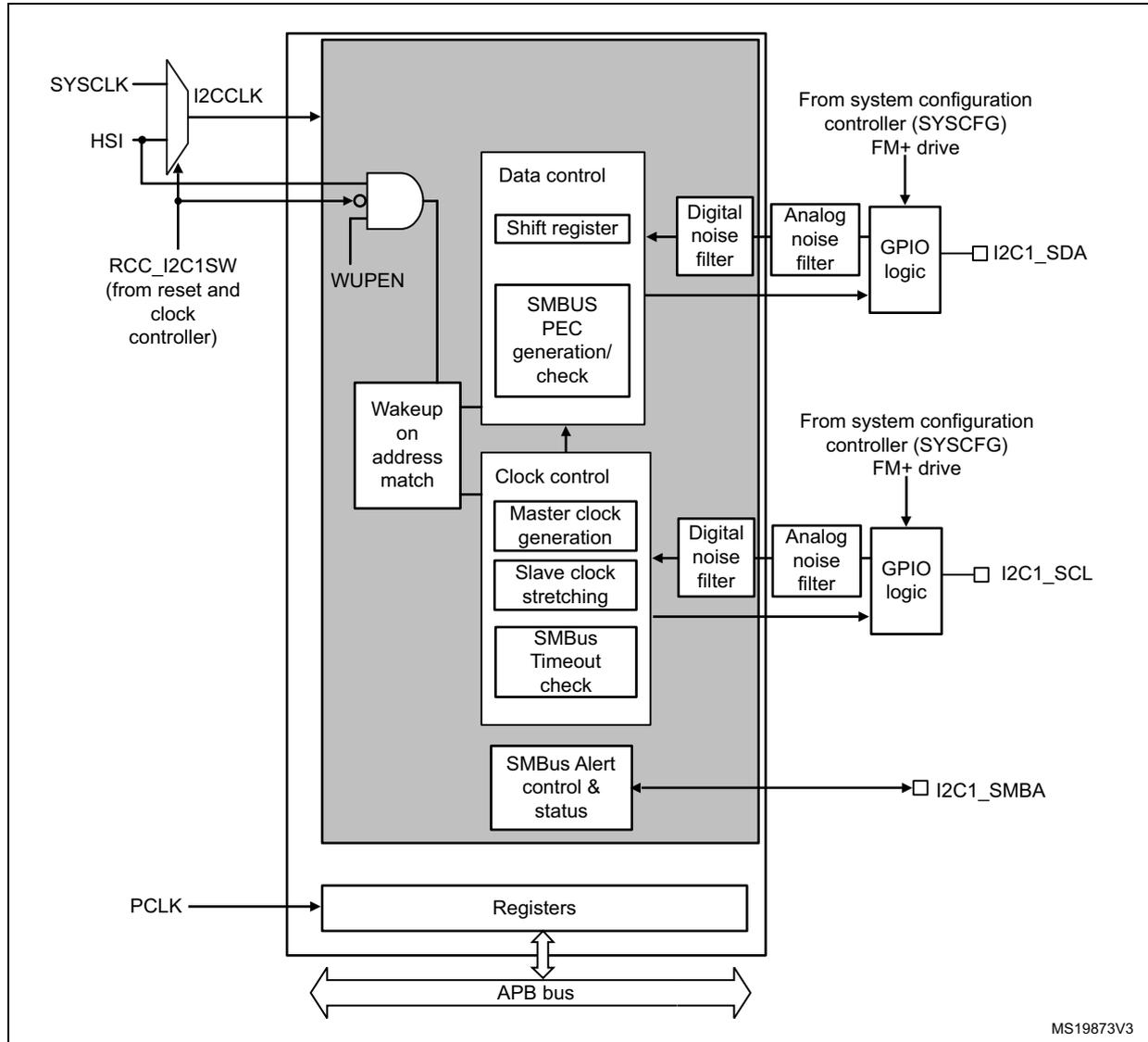
This interface can also be connected to a SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported: the additional optional SMBus Alert pin (SMBA) is also available.

### 24.4.1 I2C block diagram

The block diagram of the I2C interface is shown in [Figure 166](#).

**Figure 166. I2C block diagram**



The I2C is clocked by an independent clock source which allows to the I2C to operate independently from the PCLK frequency.

This independent clock source can be selected for either of the following two clock sources:

- HSI: high speed internal oscillator (default value)
- SYSCLK: system clock

Refer to [Section 7: Reset and clock control \(RCC\)](#) for more details.

I2C I/Os support 20 mA output current drive for Fast-mode Plus operation. This is enabled by setting the driving capability control bits for SCL and SDA in [Section 9.1.1: SYSCFG configuration register 1 \(SYSCFG\\_CFGR1\)](#).

### 24.4.2 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period  $t_{I2CCLK}$  must respect the following conditions:

$$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4 \text{ and } t_{I2CCLK} < t_{HIGH}$$

with:

$t_{LOW}$ : SCL low time and  $t_{HIGH}$ : SCL high time

$t_{filters}$ : when enabled, sum of the delays brought by the analog filter and by the digital filter.

Analog filter delay is maximum 260 ns. Digital filter delay is  $DNF \times t_{I2CCLK}$ .

The PCLK clock period  $t_{PCLK}$  must respect the following condition:

$$t_{PCLK} < 4/3 t_{SCL}$$

with  $t_{SCL}$ : SCL period

**Caution:** When the I2C kernel is clocked by PCLK. PCLK must respect the conditions for  $t_{I2CCLK}$ .

### 24.4.3 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

#### Communication flow

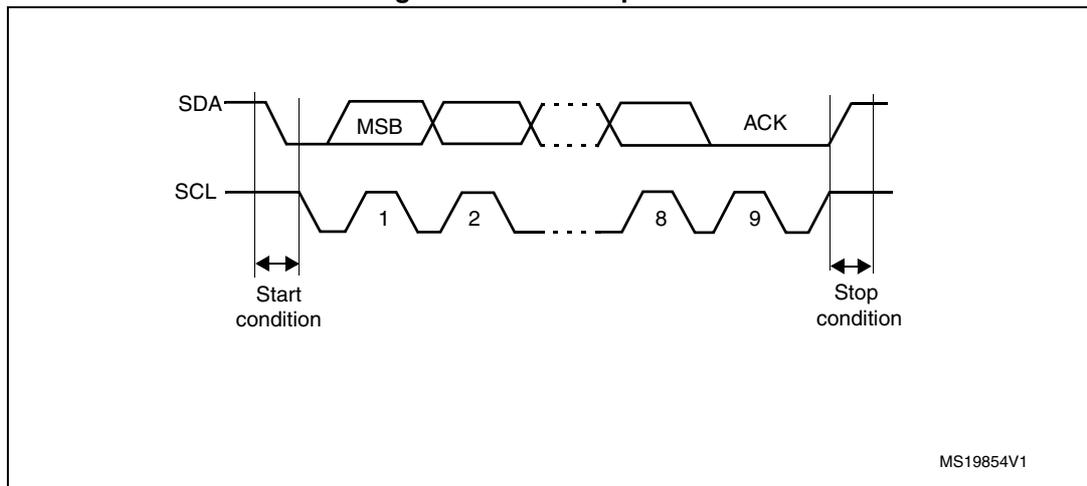
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

Figure 167. I<sup>2</sup>C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

## 24.4.4 I2C initialization

### Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller (refer to [Section 7: Reset and clock control \(RCC\)](#)).

Then the I2C can be enabled by setting the PE bit in the I2Cx\_CR1 register.

When the I2C is disabled (PE=0), the I<sup>2</sup>C performs a software reset. Refer to [Section 24.4.5: Software reset](#) for more details.

### Noise filters

Before you enable the I2C peripheral by setting the PE bit in I2Cx\_CR1 register, you must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I<sup>2</sup>C specification which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. You can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2Cx\_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x I2CCLK periods. This allows to suppress spikes with a programmable length of 1 to 15 I2CCLK periods.

**Table 68. Comparison of analog vs. digital filters**

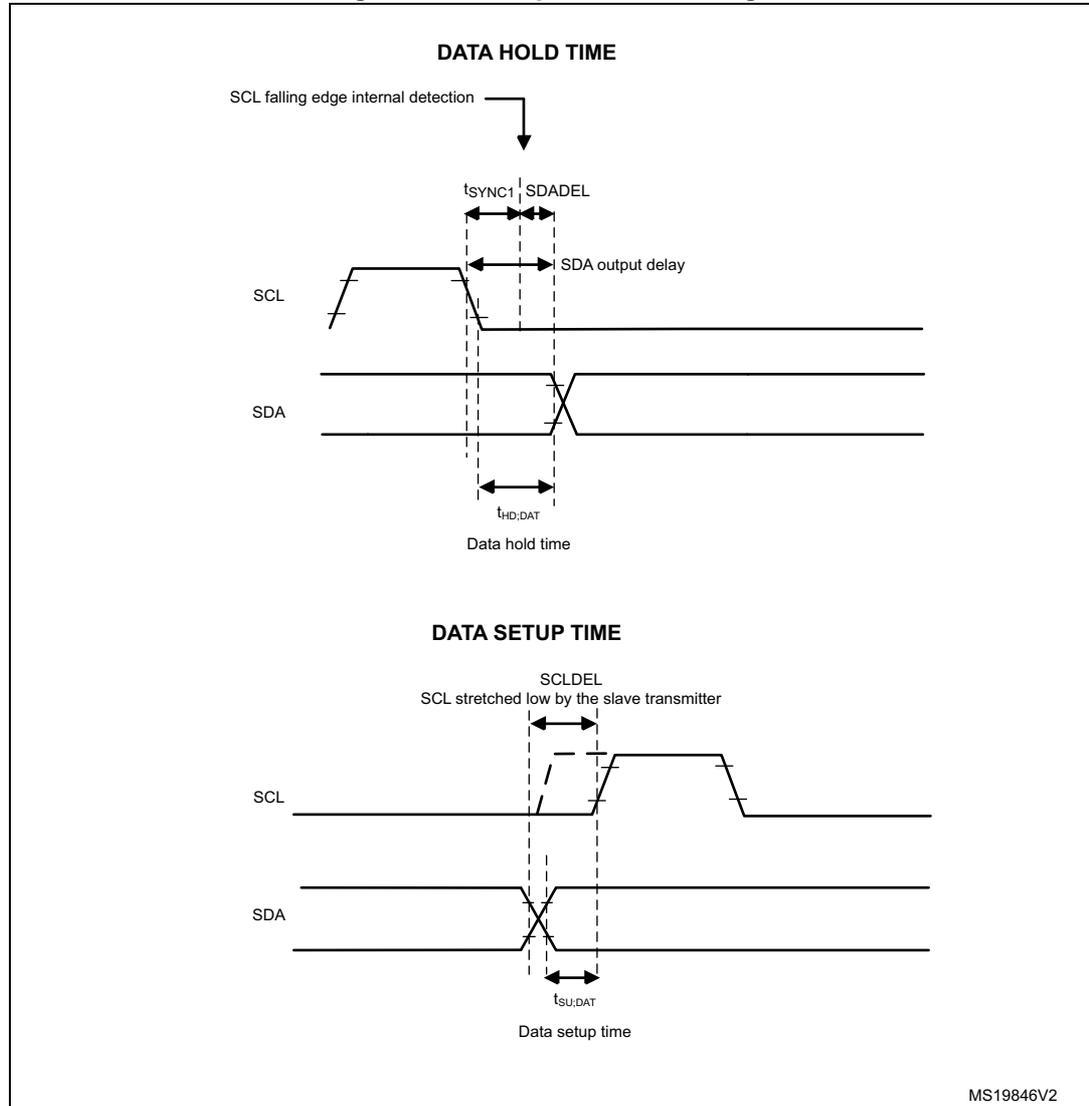
	Analog filter	Digital filter
Pulse width of suppressed spikes	≥ 50 ns	Programmable length from 1 to 15 I2C peripheral clocks
Benefits	Available in Stop mode	<ul style="list-style-type: none"> <li>– Programmable length: extra filtering capability vs. standard requirements</li> <li>– Stable length</li> </ul>
Drawbacks	Variation vs. temperature, voltage, process	Wakeup from Stop mode on address match is not available when digital filter is enabled

**Caution:** Changing the filter configuration is not allowed when the I2C is enabled.

### I2C timings

The timings must be configured in order to guarantee a correct data hold and setup time, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2Cx\_TIMINGR register.

Figure 168. Setup and hold timings



- When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is  $t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}$  where  $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$ .  
 $t_{SDADEL}$  impacts the hold time  $t_{HD,DAT}$ .

The total SDA output delay is:

$$t_{SYNC1} + \{ [SDADEL \times (PRESC+1) + 1] \times t_{I2CCLK} \}$$

$t_{\text{SYNC1}}$  duration depends on these parameters:

- SCL falling slope
- When enabled, input delay brought by the analog filter:  $t_{\text{AF}(\text{min})} < t_{\text{AF}} < t_{\text{AF}(\text{max})}$  ns.
- When enabled, input delay brought by the digital filter:  $t_{\text{DNF}} = \text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization to I2CCLK clock (2 to 3 I2CCLK periods)

In order to bridge the undefined region of the SCL falling edge, you must program SDADEL in such a way that:

$$\{t_{\text{f}(\text{max})} + t_{\text{HD;DAT}(\text{min})} - t_{\text{AF}(\text{min})} - [(DNF + 3) \times t_{\text{I2CCLK}}]\} / \{(\text{PRESC} + 1) \times t_{\text{I2CCLK}}\} \leq \text{SDADEL}$$

$$\text{SDADEL} \leq \{t_{\text{HD;DAT}(\text{max})} - t_{\text{AF}(\text{max})} - [(DNF + 4) \times t_{\text{I2CCLK}}]\} / \{(\text{PRESC} + 1) \times t_{\text{I2CCLK}}\}$$

**Note:**  $t_{\text{AF}(\text{min})} / t_{\text{AF}(\text{max})}$  are part of the equation only when the analog filter is enabled. Refer to device datasheet for  $t_{\text{AF}}$  values.

The maximum  $t_{\text{HD;DAT}}$  could be 3.45  $\mu\text{s}$ , 0.9  $\mu\text{s}$  and 0.45  $\mu\text{s}$  for Standard-mode, Fast-mode and Fast-mode Plus, but must be less than the maximum of  $t_{\text{VD;DAT}}$  by a transition time. This maximum must only be met if the device does not stretch the LOW period ( $t_{\text{LOW}}$ ) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case, so in this case the previous equation becomes:

$$\text{SDADEL} \leq \{t_{\text{VD;DAT}(\text{max})} - t_{\text{r}(\text{max})} - 260 \text{ ns} - [(DNF + 4) \times t_{\text{I2CCLK}}]\} / \{(\text{PRESC} + 1) \times t_{\text{I2CCLK}}\}.$$

**Note:** This condition can be violated when  $\text{NOSTRETCH}=0$ , because the device stretches SCL low to guarantee the set-up time, according to the  $\text{SCLDEL}$  value.

Refer to [Table 69: I2C-SMBUS specification data setup and hold times](#) for  $t_{\text{f}}$ ,  $t_{\text{r}}$ ,  $t_{\text{HD;DAT}}$  and  $t_{\text{VD;DAT}}$  standard values.

- After sending SDA output, SCL line is kept at low level during the setup time. This setup time is  $t_{\text{SCLDEL}} = (\text{SCLDEL} + 1) \times t_{\text{PRESC}}$  where  $t_{\text{PRESC}} = (\text{PRESC} + 1) \times t_{\text{I2CCLK}}$ .  
 $t_{\text{SCLDEL}}$  impacts the setup time  $t_{\text{SU;DAT}}$ .

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), you must program SCLDEL in such a way that:

$$\{[t_{\text{r}(\text{max})} + t_{\text{SU;DAT}(\text{min})}] / [(\text{PRESC} + 1) \times t_{\text{I2CCLK}}]\} - 1 \leq \text{SCLDEL}$$

Refer to [Table 69: I2C-SMBUS specification data setup and hold times](#) for  $t_{\text{r}}$  and  $t_{\text{SU;DAT}}$  standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Table 69. I2C-SMBUS specification data setup and hold times

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		SMBUS		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0		0		0		0.3		$\mu\text{s}$
$t_{VD;DAT}$	Data valid time		3.45		0.9		0.45			
$t_{SU;DAT}$	Data setup time	250		100		50		250		ns
$t_r$	Rise time of both SDA and SCL signals		1000		300		120		1000	
$t_f$	Fall time of both SDA and SCL signals		300		300		120		300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2Cx\_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is  $t_{SCLL} = (SCLL+1) \times t_{PRESC}$  where  $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$ .  
 $t_{SCLL}$  impacts the SCL low time  $t_{LOW}$ .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is  $t_{SCLH} = (SCLH+1) \times t_{PRESC}$  where  $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$ .  $t_{SCLH}$  impacts the SCL high time  $t_{HIGH}$ .

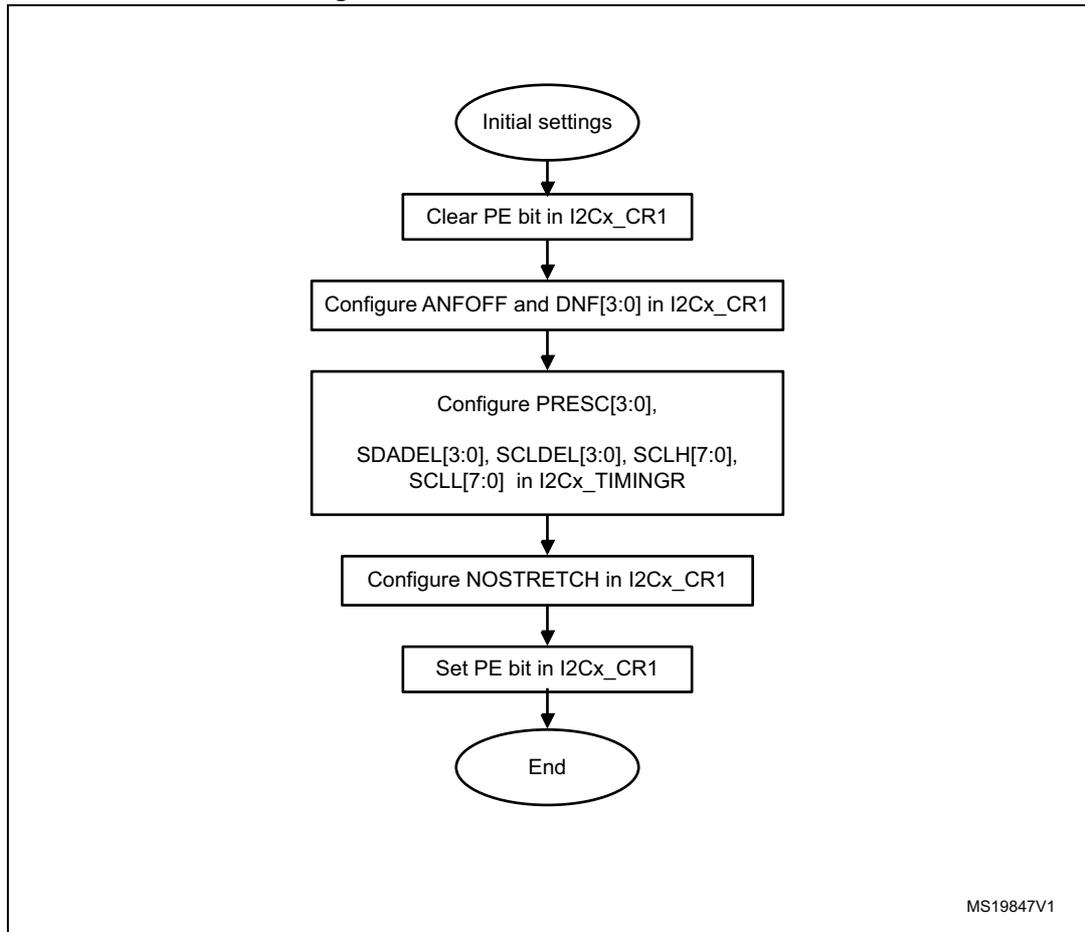
Refer to section : [I2C master initialization](#) for more details.

**Caution:** Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to : [I2C slave initialization](#) for more details.

**Caution:** Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 169. I2C initialization flowchart



#### 24.4.5 Software reset

A software reset can be performed by clearing the PE bit in the I2Cx\_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2Cx\_CR2 register: START, STOP, NACK
2. I2Cx\_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2Cx\_CR2 register: PECBYTE
2. I2Cx\_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least 3 APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence: - Write PE=0 - Check PE=0 - Write PE=1

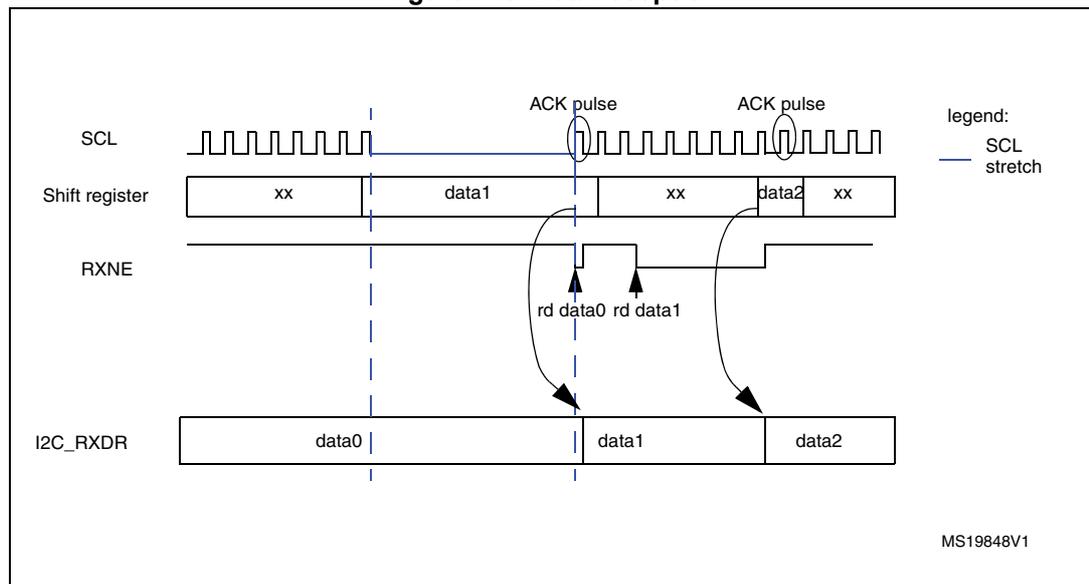
### 24.4.6 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

#### Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2Cx\_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2Cx\_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

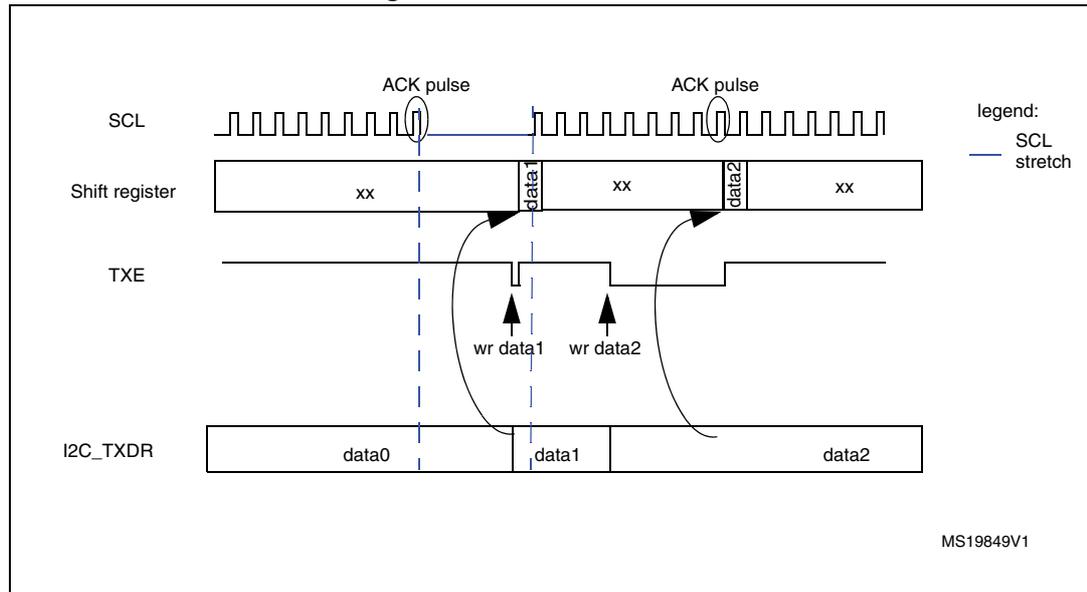
Figure 170. Data reception



#### Transmission

If the I2Cx\_TXDR register is not empty (TXE=0), its content is copied into the shift register after the 9th SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE=1, meaning that no data is written yet in I2Cx\_TXDR, SCL line is stretched low until I2Cx\_TXDR is written. The stretch is done after the 9th SCL pulse.

Figure 171. Data transmission



### Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (Slave Byte Control) bit in the I2Cx\_CR2 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2Cx\_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2Cx\_CR2 register. In this mode, TCR flag is set when the number of bytes programmed in NBYTES has been transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in 2 modes:

- **Automatic end mode** (AUTOEND = '1' in the I2Cx\_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred.
- **Software end mode** (AUTOEND = '0' in the I2Cx\_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field has been transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2Cx\_CR2 register. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

**Table 70. I2C Configuration table**

Function	SBC bit	RELOAD bit	AUTOEND bit
Master Tx/Rx NBYTES + STOP	x	0	1
Master Tx/Rx + NBYTES + RESTART	x	0	0
Slave Tx/Rx all received bytes ACKed	0	x	x
Slave Rx with ACK control	1	1	x

## 24.4.7 I2C slave mode

### I2C slave initialization

In order to work in slave mode, you must enable at least one slave address. Two registers I2Cx\_OAR1 and I2Cx\_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2Cx\_OAR1 register.  
OA1 is enabled by setting the OA1EN bit in the I2Cx\_OAR1 register.
- If additional slave addresses are required, you can configure the 2nd slave address OA2. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2Cx\_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.  
These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2Cx\_OAR1 or I2Cx\_OAR2 register with OA2MSK=0.  
OA2 is enabled by setting the OA2EN bit in the I2Cx\_OAR2 register.
- The General Call address is enabled by setting the GCEN bit in the I2Cx\_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH=1 in the I2Cx\_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled you must read the ADDCODE[6:0] bits in the I2Cx\_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

### Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2Cx\_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE=1). This stretch is released when the data is written to the I2Cx\_TXDR register.
- In reception when the I2Cx\_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2Cx\_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during  $[(SDADEL+SCLDEL+1) \times (PRESC+1) + 1] \times t_{I2CCLK}$ .

### Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2Cx\_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2Cx\_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2Cx\_ISR register and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if you clear the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, you ensure that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2Cx\_RXDR register before the 9th SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2Cx\_ISR register and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

### Slave Byte Control Mode

In order to allow byte ACK control in slave reception mode, Slave Byte Control mode must be enabled by setting the SBC bit in the I2Cx\_CR1 register. This is required to be compliant with SMBus standards.

Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD=1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the 8th and 9th SCL pulses. You can read the data from the I2Cx\_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2Cx\_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

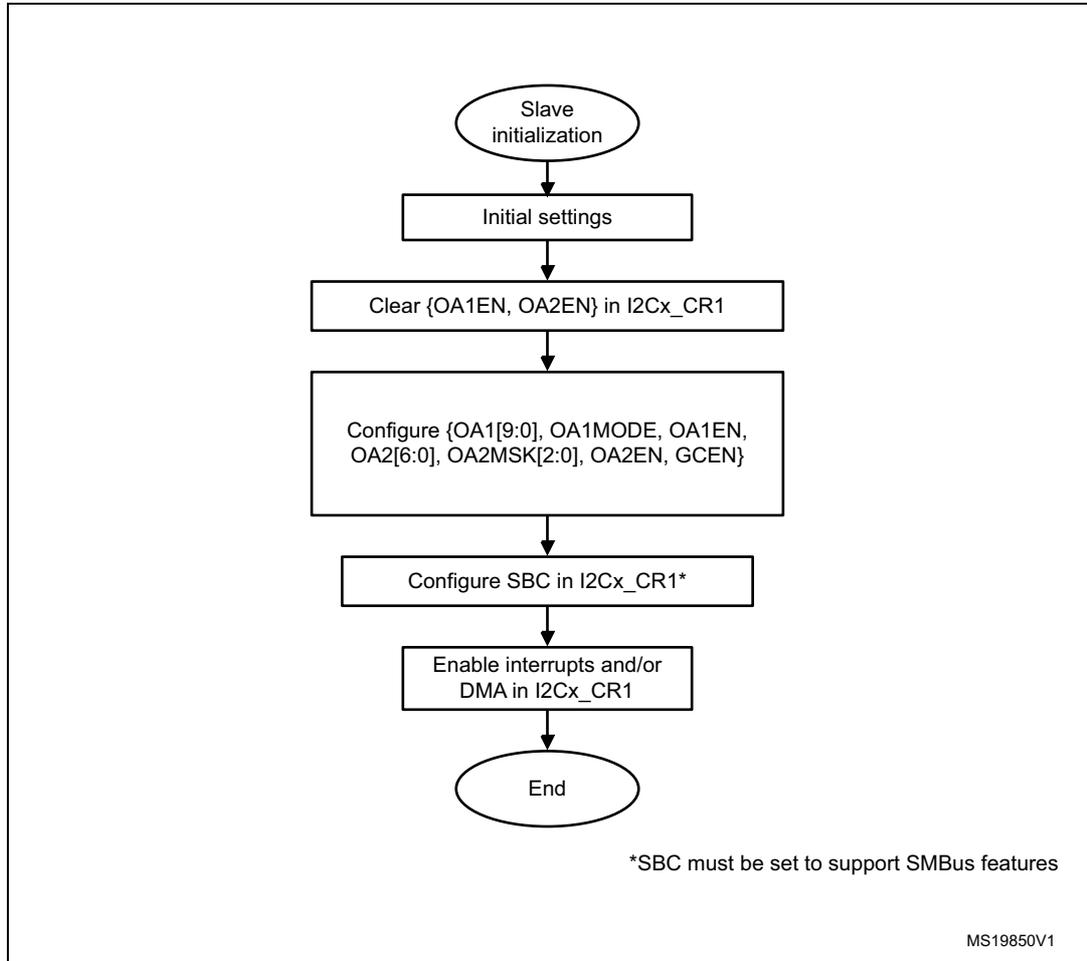
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

**Note:** The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR=1.

The RELOAD bit value can be changed when ADDR=1, or when TCR=1.

**Caution:** Slave Byte Control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH=1 is not allowed.

**Figure 172. Slave initialization flowchart**



**Slave transmitter**

A transmit interrupt status (TXIS) is generated when the I2Cx\_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2Cx\_CR1 register.

The TXIS bit is cleared when the I2Cx\_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2Cx\_ISR register and an interrupt is generated if the NACKIE bit is set in the I2Cx\_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2Cx\_CR1 register, the STOPF flag is set in the I2Cx\_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, if TXE = 0 when the slave address is received (ADDR=1), you can choose either to send the content of the I2Cx\_TXDR register as the first data byte, or to flush the I2Cx\_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave Byte Control mode (SBC=1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR=1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

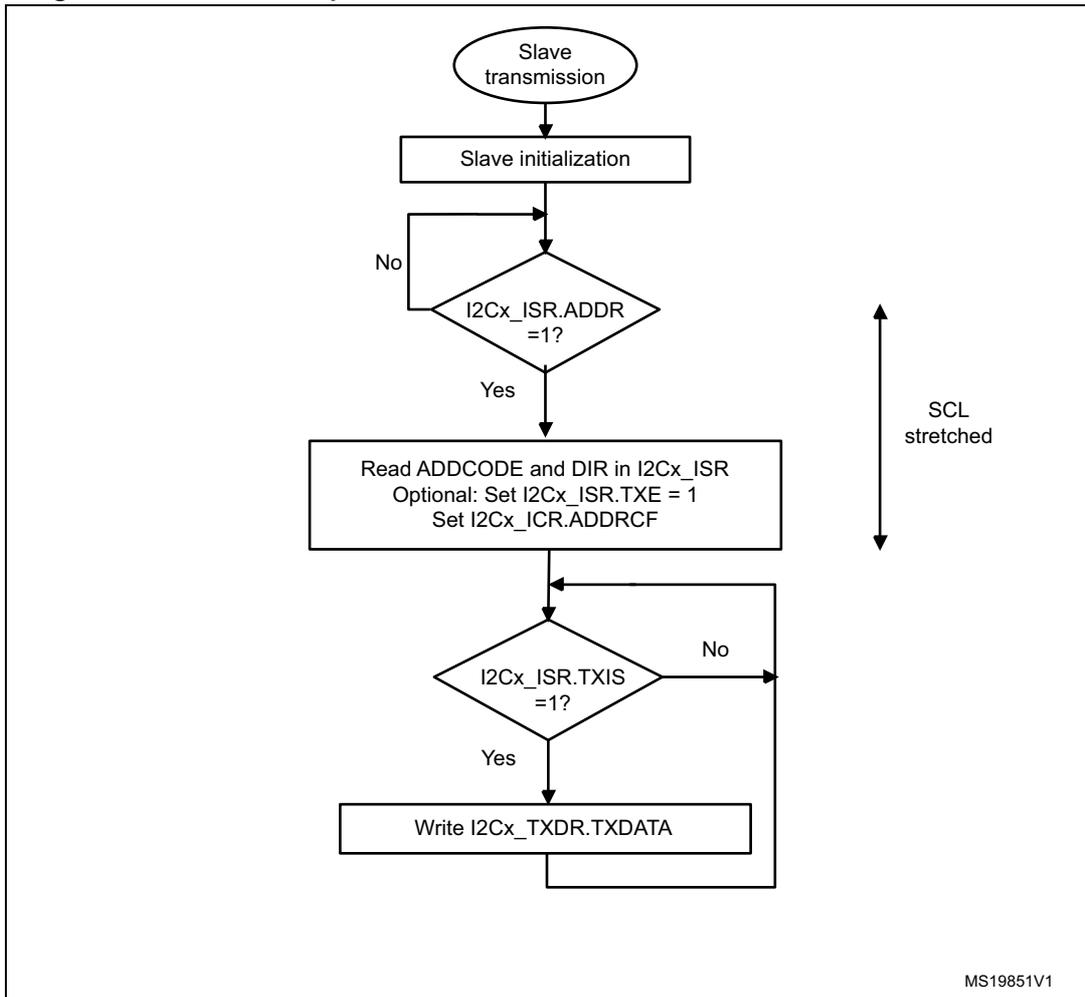
**Caution:** When NOSTRETCH=1, the SCL clock is not stretched while the ADDR flag is set, so you cannot flush the I2Cx\_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2Cx\_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2Cx\_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error will be generated (the OVR flag is set).

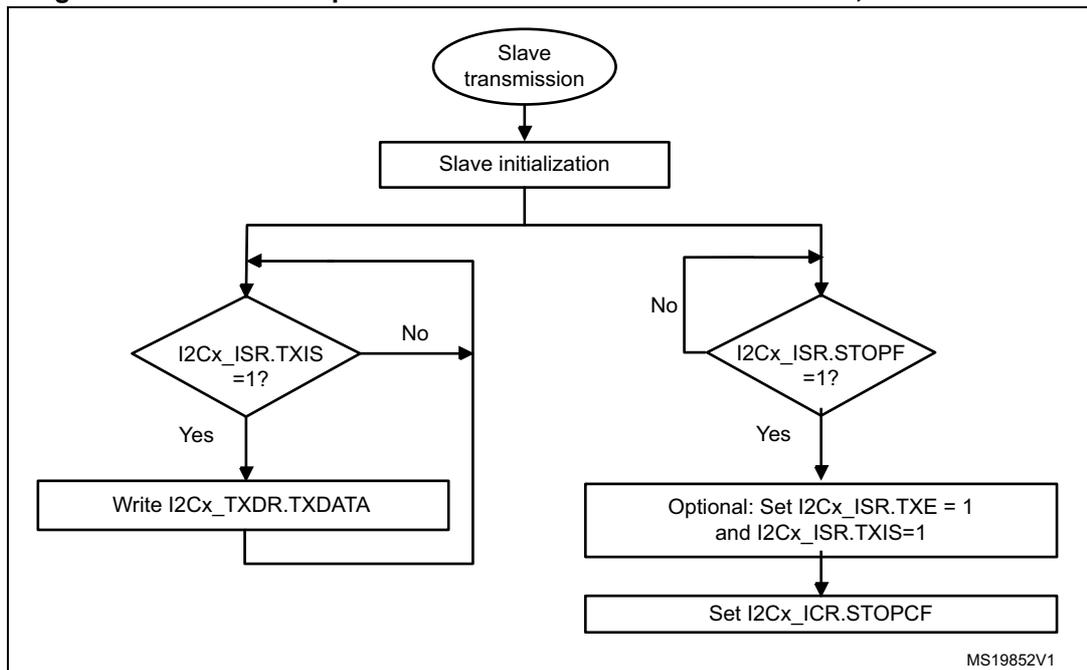
If you need a TXIS event, (Transmit Interrupt or Transmit DMA request), you must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 173. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=0



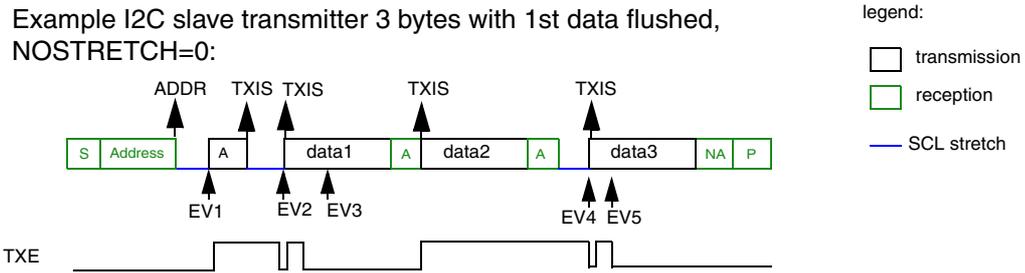
MS19851V1

Figure 174. Transfer sequence flowchart for I2C slave transmitter, NOSTRETCH=1

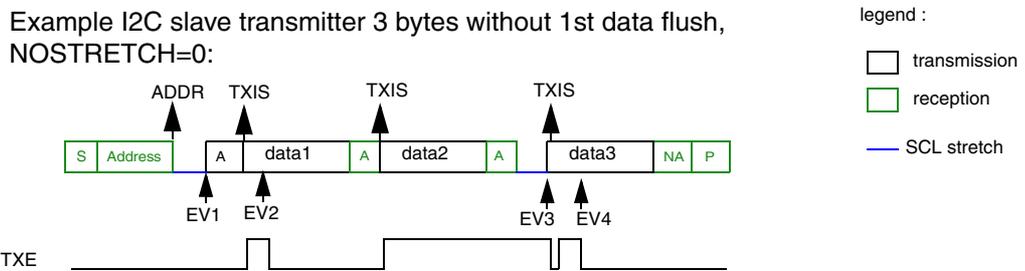


MS19852V1

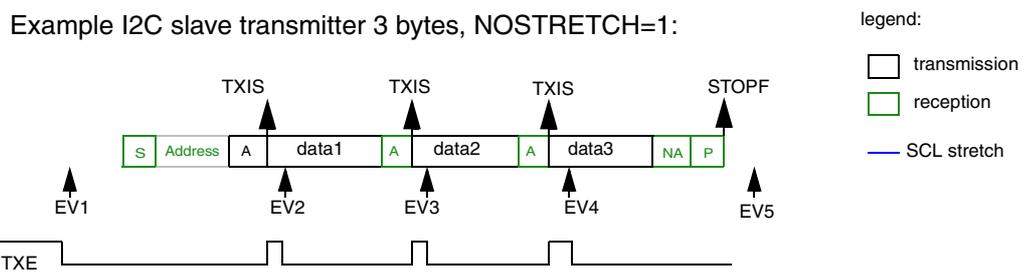
Figure 175. Transfer bus diagrams for I2C slave transmitter



EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF  
 EV2: TXIS ISR: wr data1  
 EV3: TXIS ISR: wr data2  
 EV4: TXIS ISR: wr data3  
 EV5: TXIS ISR: wr data4 (not sent)



EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF  
 EV2: TXIS ISR: wr data2  
 EV3: TXIS ISR: wr data3  
 EV4: TXIS ISR: wr data4 (not sent)



EV1: wr data1  
 EV2: TXIS ISR: wr data2  
 EV3: TXIS ISR: wr data3  
 EV4: TXIS ISR: wr data4 (not sent)  
 EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

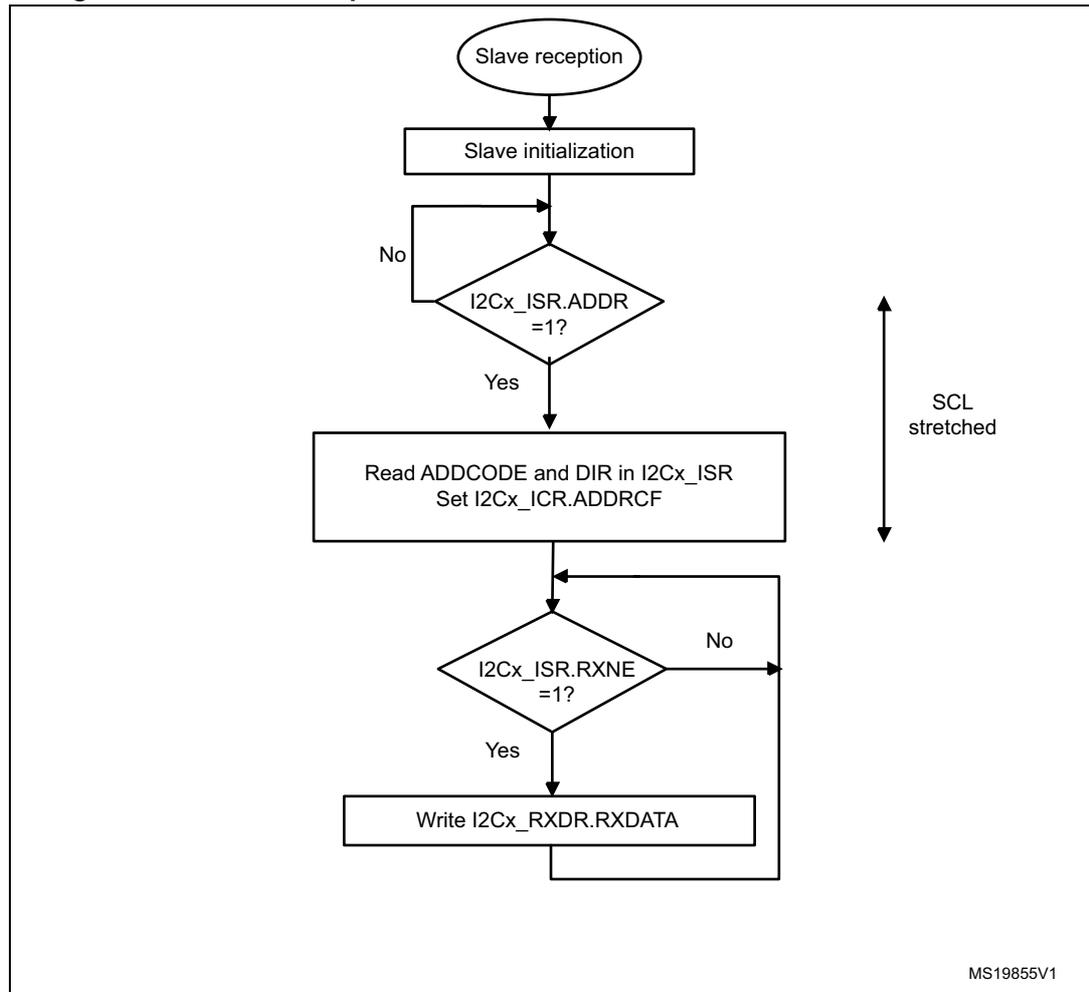
MS19853V1

### Slave receiver

RXNE is set in I2Cx\_ISR when the I2Cx\_RXDR is full, and generates an interrupt if RXIE is set in I2Cx\_CR1. RXNE is cleared when I2Cx\_RXDR is read.

When a STOP is received and STOPIE is set in I2Cx\_CR1, STOPF is set in I2Cx\_ISR and an interrupt is generated.

Figure 176. Transfer sequence flowchart for slave receiver with NOSTRETCH=0



MS19855V1

Figure 177. Transfer sequence flowchart for slave receiver with NOSTRETCH=1

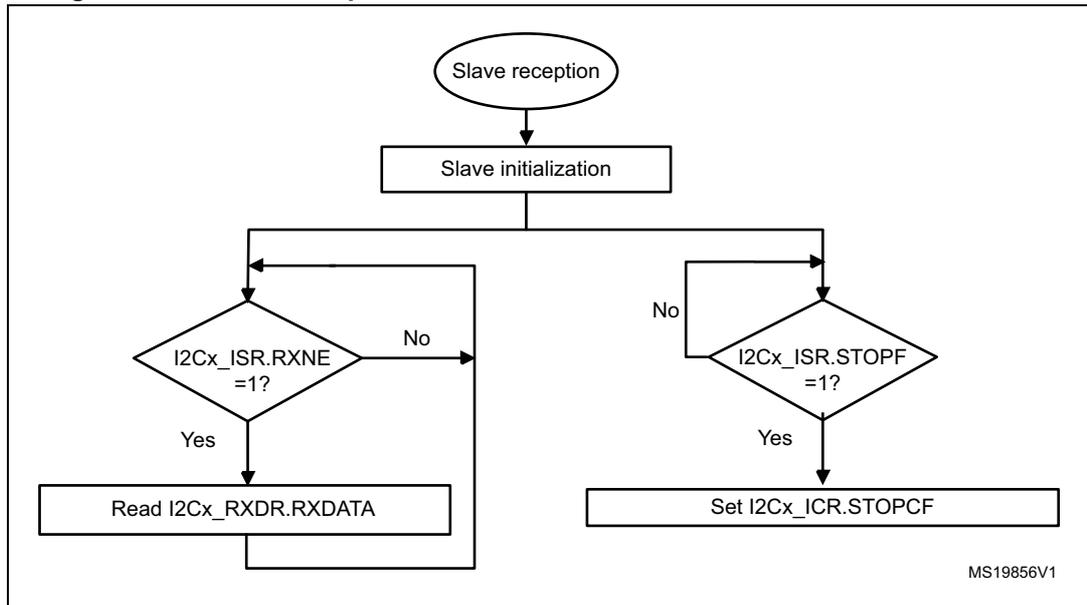
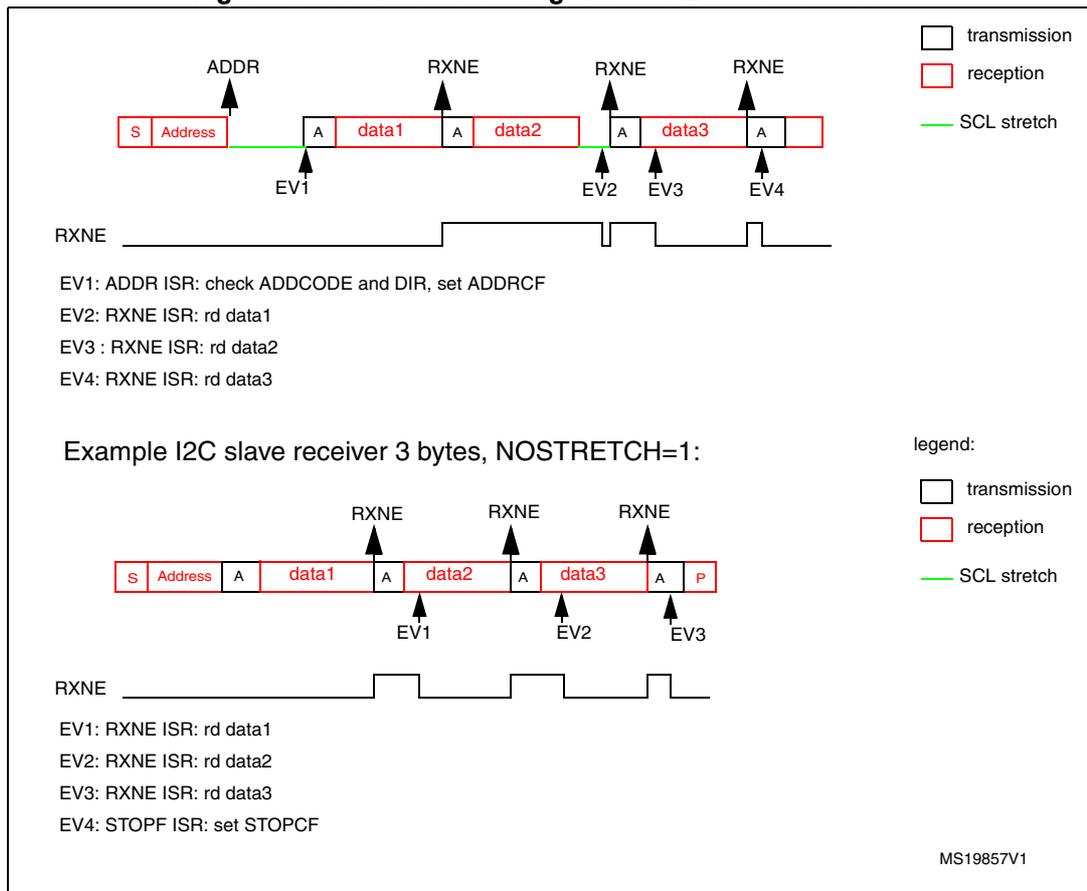


Figure 178. Transfer bus diagrams for I2C slave receiver



## 24.4.8 I2C master mode

### I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2Cx\_TIMINGR register.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a  $t_{\text{SYNC1}}$  delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2Cx\_TIMINGR register.

The I2C detects its own SCL high level after a  $t_{\text{SYNC2}}$  delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2Cx\_TIMINGR register.

Consequently the master clock period is:

$$t_{\text{SCL}} = t_{\text{SYNC1}} + t_{\text{SYNC2}} + \{ [(SCLH+1) + (SCLL+1)] \times (\text{PRESC}+1) \times t_{\text{I2CCLK}} \}$$

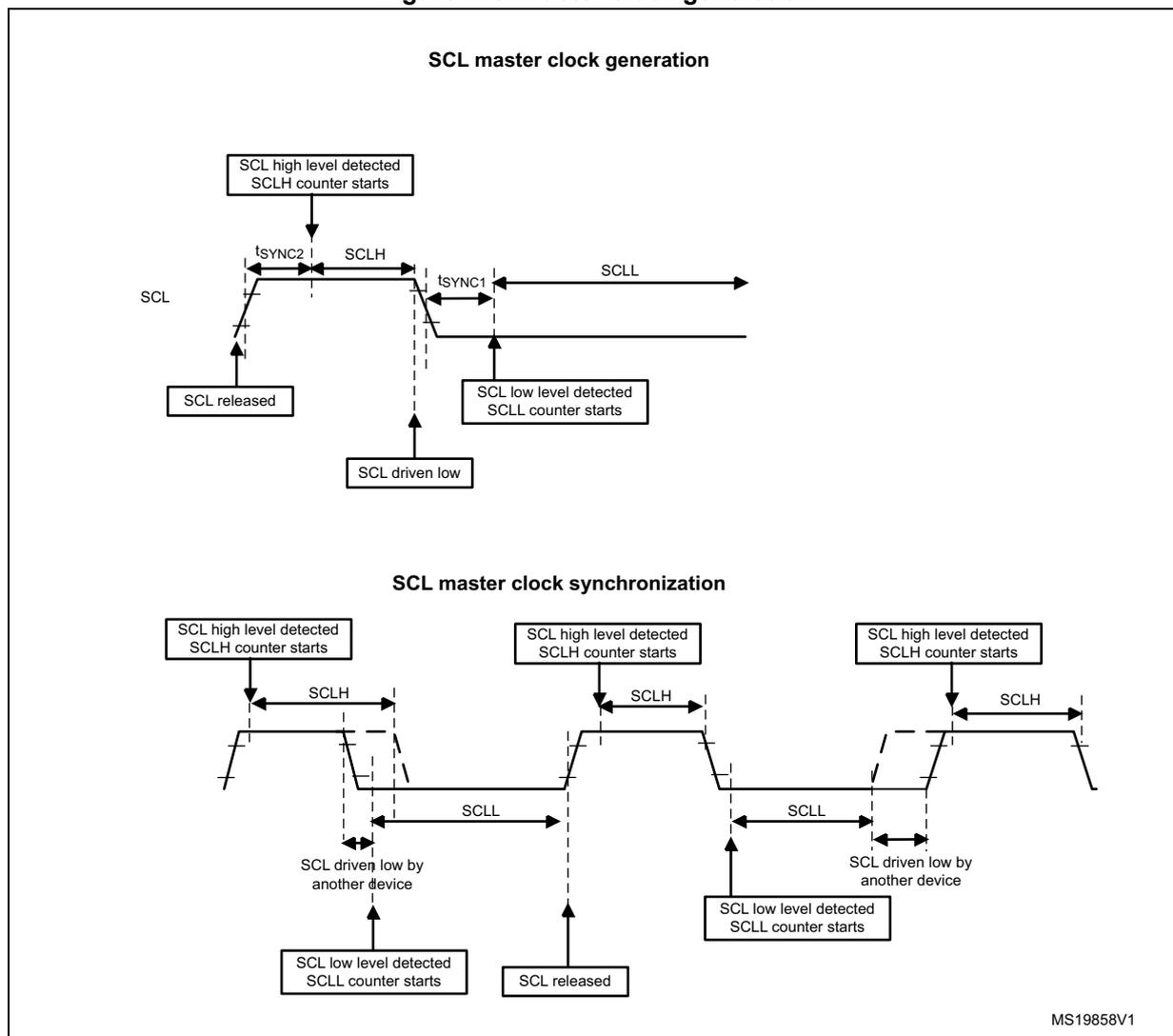
The duration of  $t_{\text{SYNC1}}$  depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter:  $\text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

The duration of  $t_{\text{SYNC2}}$  depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter:  $\text{DNF} \times t_{\text{I2CCLK}}$
- Delay due to SCL synchronization with I2CCLK clock (2 to 3 I2CCLK periods)

Figure 179. Master clock generation



MS19858V1

**Caution:** In order to be I<sup>2</sup>C or SMBus compliant, the master clock must respect the timings given below:

Table 71. I2C-SMBUS specification clock timings

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f <sub>SCL</sub>	SCL clock frequency		100		400		1000		100	kHz
t <sub>HD:STA</sub>	Hold time (repeated) START condition	4.0		0.6		0.26		4.0		µs
t <sub>SU:STA</sub>	Set-up time for a repeated START condition	4.7		0.6		0.26		4.7		µs
t <sub>SU:STO</sub>	Set-up time for STOP condition	4.0		0.6		0.26		4.0		µs

Table 71. I2C-SMBUS specification clock timings (continued)

Symbol	Parameter	Standard-mode		Fast-mode		Fast-mode Plus		SMBUS		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{BUF}$	Bus free time between a STOP and START condition	4.7		1.3		0.5		4.7		$\mu$ s
$t_{LOW}$	Low period of the SCL clock	4.7		1.3		0.5		4.7		$\mu$ s
$t_{HIGH}$	Period of the SCL clock	4.0		0.6		0.26		4.0	50	$\mu$ s
$t_r$	Rise time of both SDA and SCL signals		1000		300		120		1000	ns
$t_f$	Fall time of both SDA and SCL signals		300		300		120		300	ns

*Note:*  $SCLL$  is also used to generate the  $t_{BUF}$  and  $t_{SU:STA}$  timings.

$SCLH$  is also used to generate the  $t_{HD:STA}$  and  $t_{SU:STO}$  timings.

Refer to [Section 24.4.9: I2Cx\\_TIMINGR register configuration examples](#) for examples of I2Cx\_TIMINGR settings vs. I2CCLK frequency.

### Master communication initialization (address phase)

In order to initiate the communication, you must program the following parameters for the addressed slave in the I2Cx\_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD\_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configured to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

You must then set the START bit in I2Cx\_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of  $t_{BUF}$ .

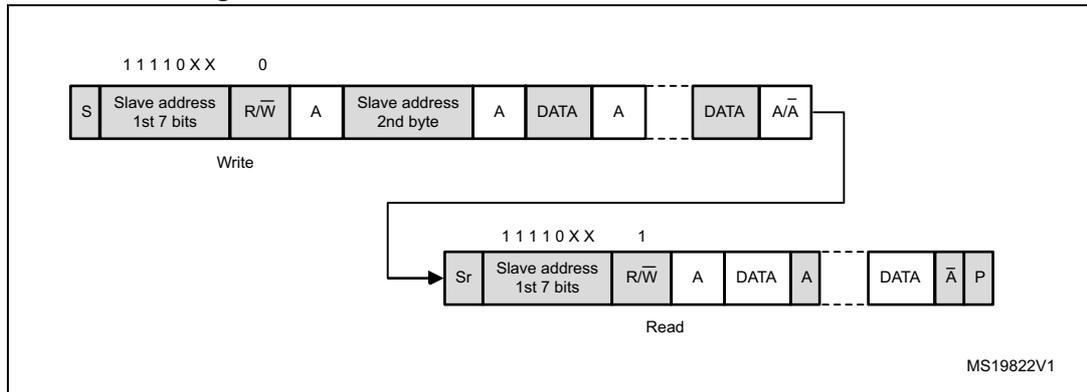
In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

*Note:* The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs. If the I2C is addressed as a slave (ADDR=1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared when the ADDRCONF bit is set.

*Note:* The same procedure is applied for a Repeated Start condition. In this case BUSY=1.



Figure 182. 10-bit address read access with HEAD10R=1



**Master transmitter**

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the 9th SCL pulse when an ACK is received.

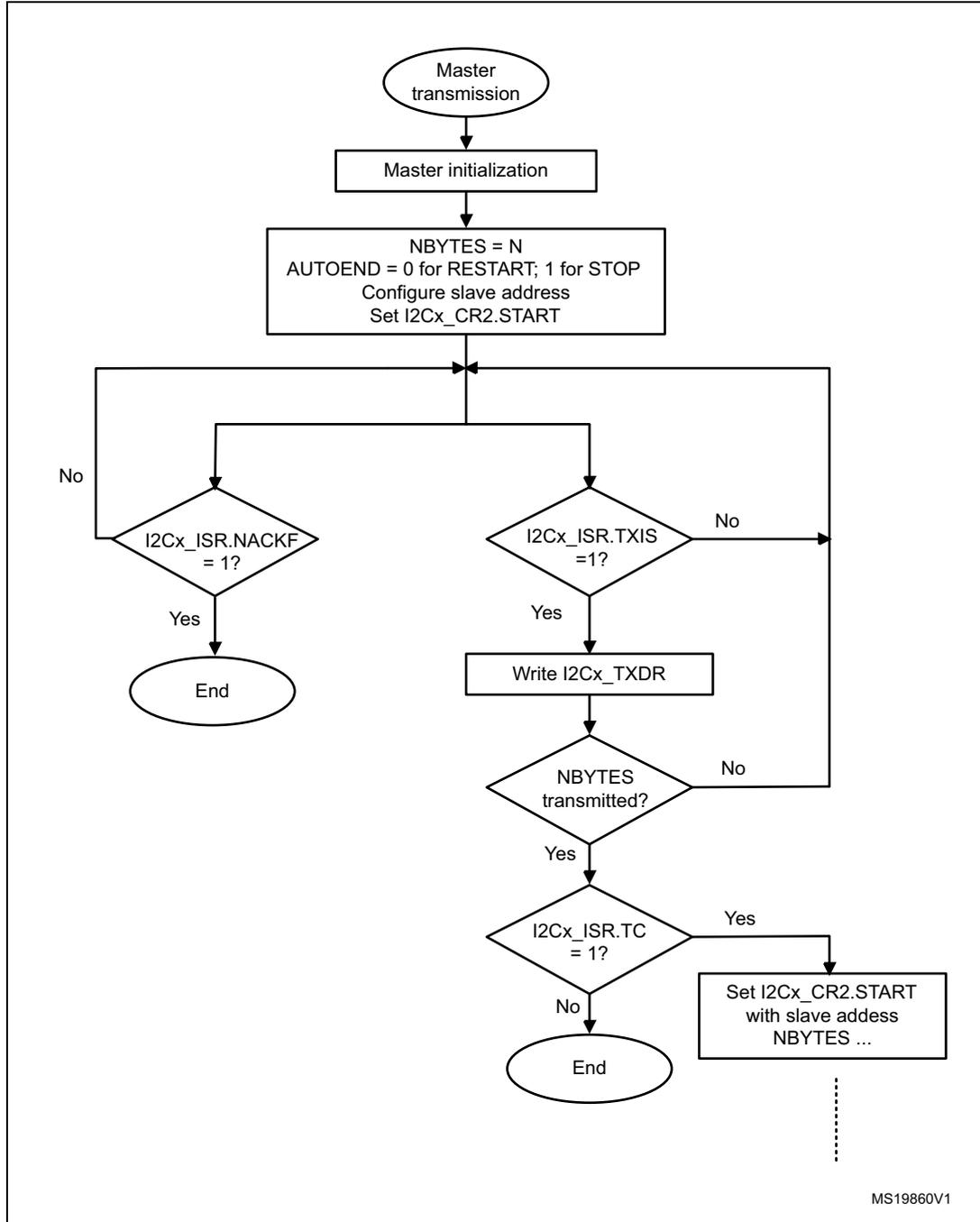
A TXIS event generates an interrupt if the TXIE bit is set in the I2Cx\_CR1 register. The flag is cleared when the I2Cx\_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2Cx\_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:
    - A RESTART condition can be requested by setting the START bit in the I2Cx\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.
    - A STOP condition can be requested by setting the STOP bit in the I2Cx\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2Cx\_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 183. Transfer sequence flowchart for I2C master transmitter for N<=255 bytes



MS19860V1

Figure 184. Transfer sequence flowchart for I2C master transmitter for N>255 bytes

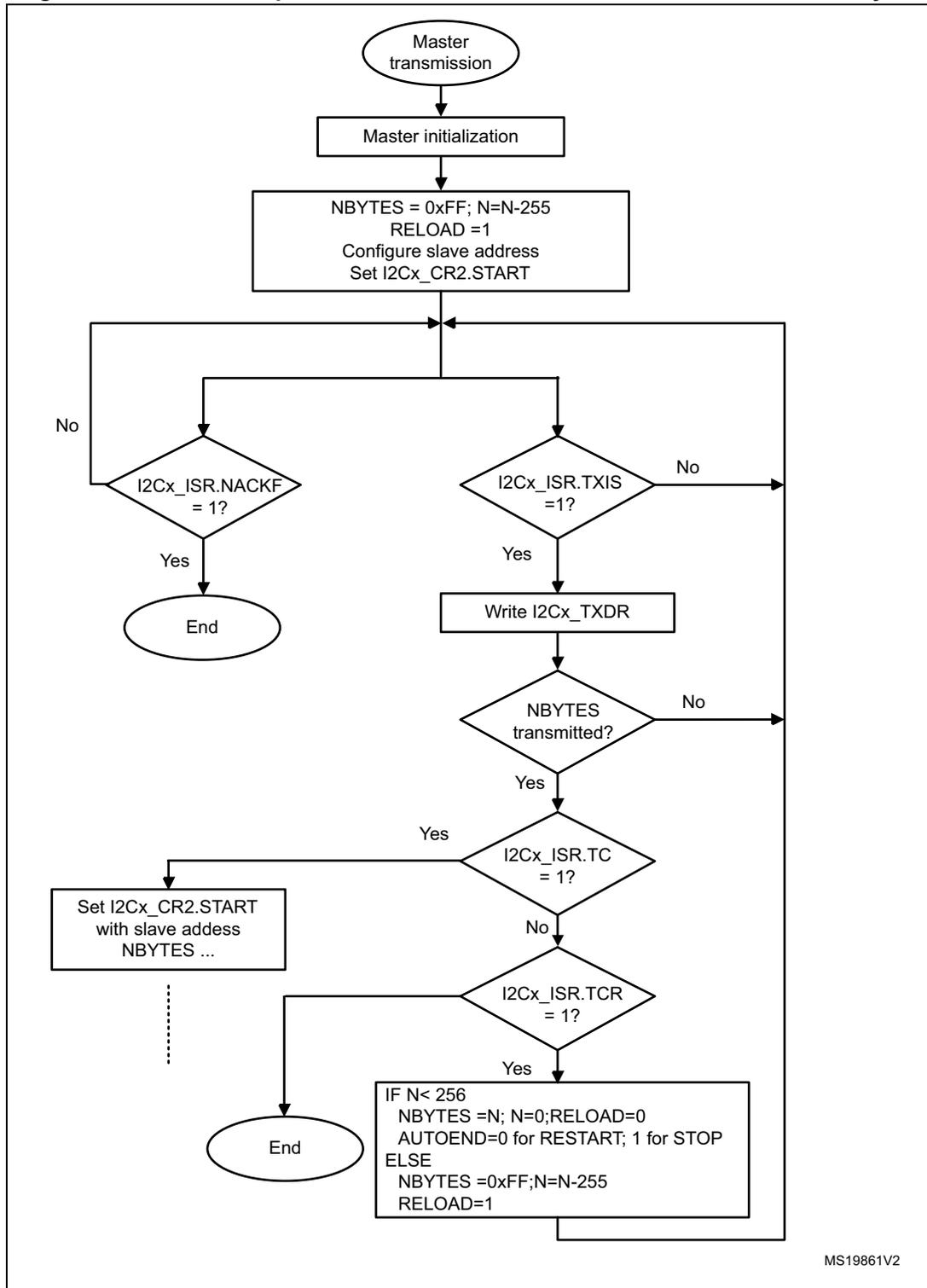
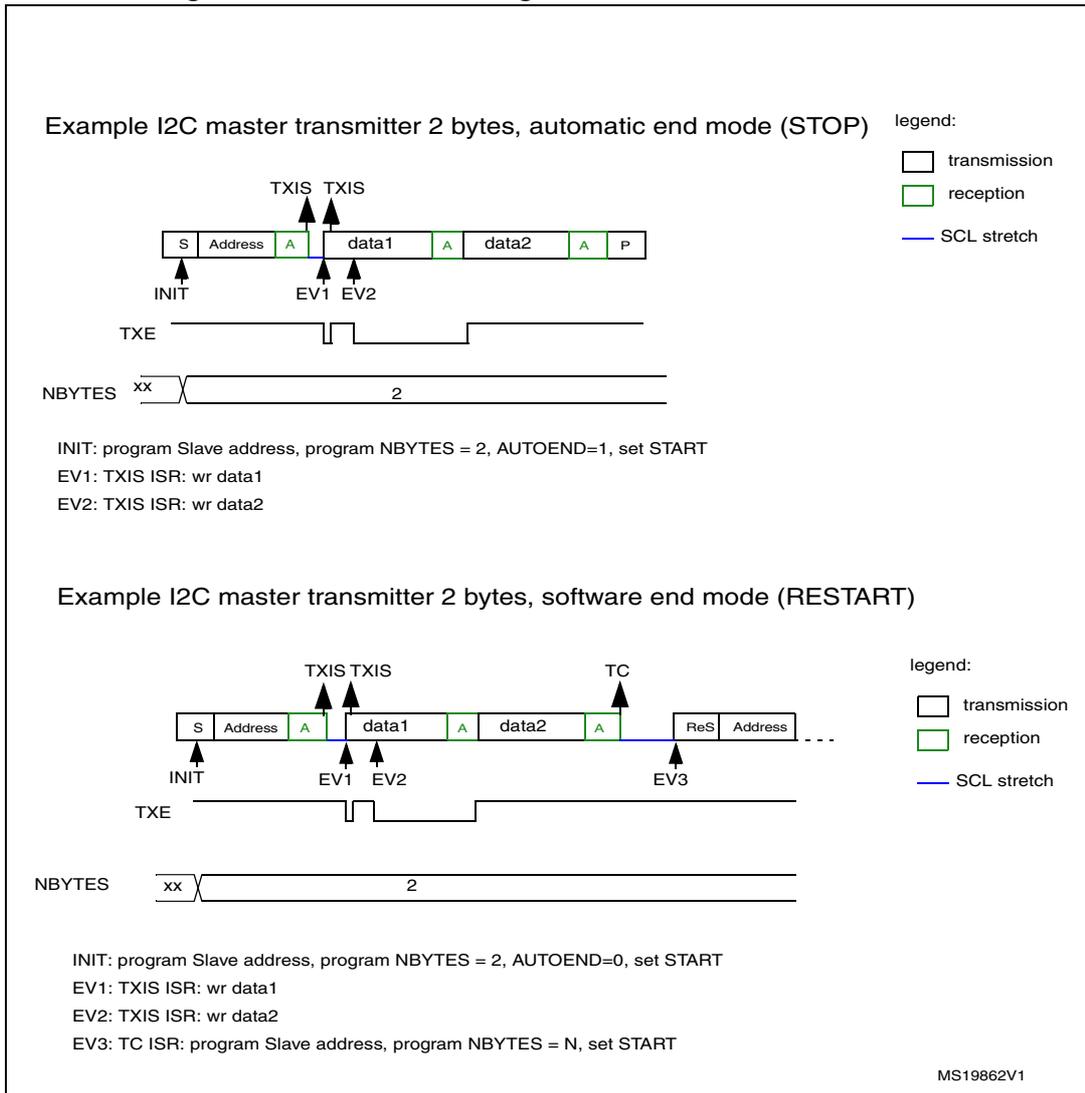


Figure 185. Transfer bus diagrams for I2C master transmitter



### Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the 8th SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2Cx\_CR1 register. The flag is cleared when I2Cx\_RXDR is read.

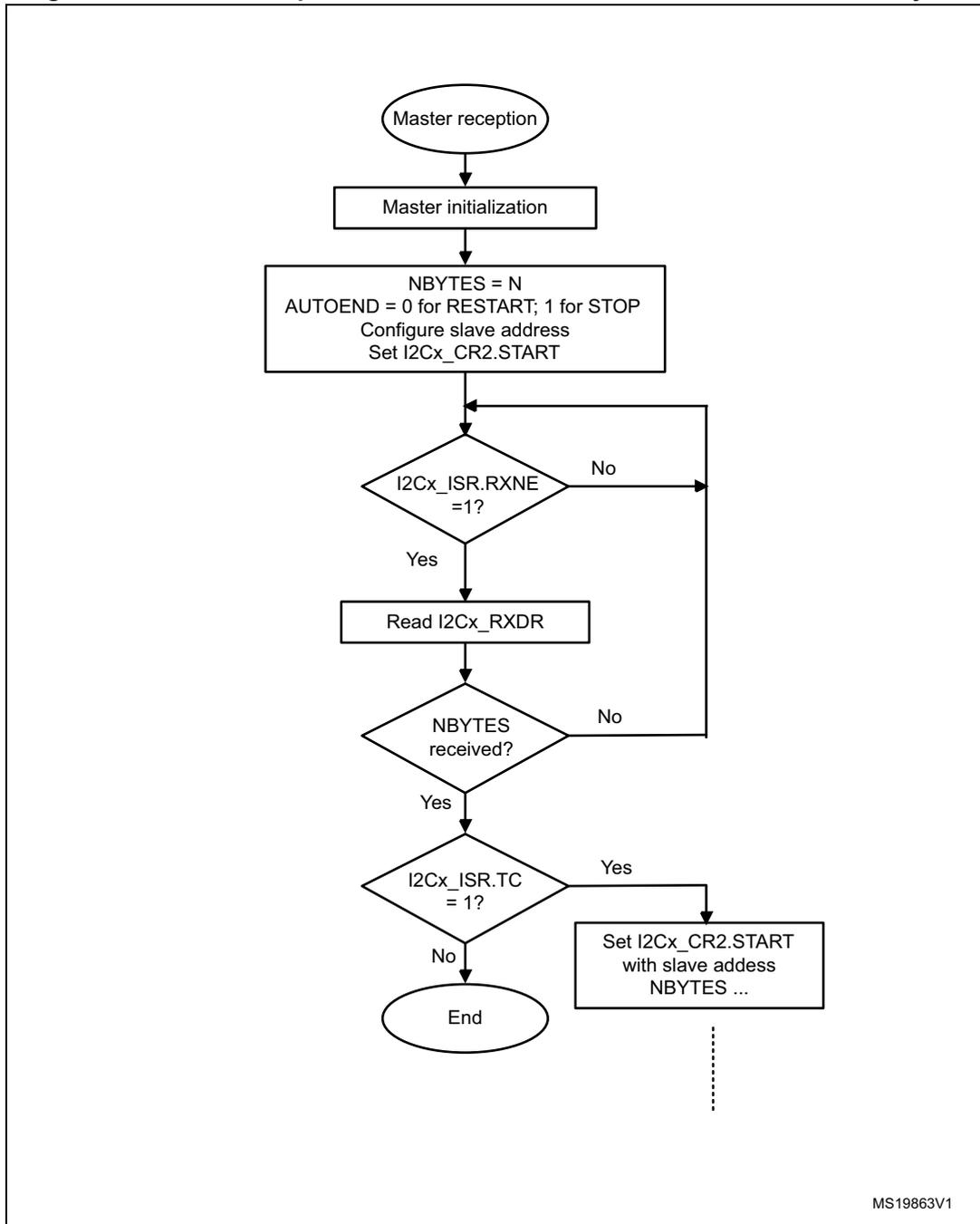
If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2Cx\_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

A RESTART condition can be requested by setting the START bit in the I2Cx\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2Cx\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 186. Transfer sequence flowchart for I2C master receiver for N<=255 bytes



MS19863V1

Figure 187. Transfer sequence flowchart for I2C master receiver for N >255 bytes

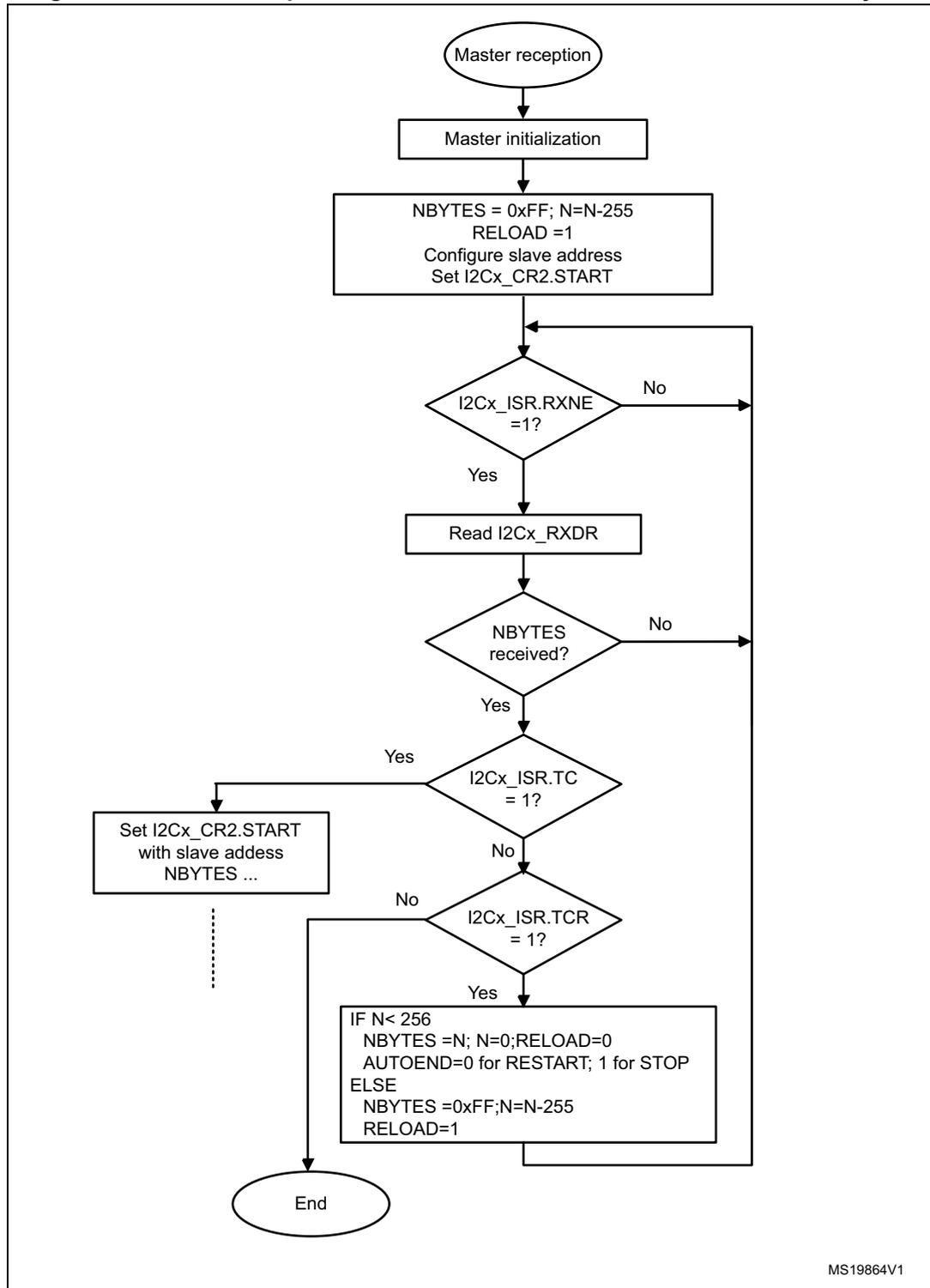
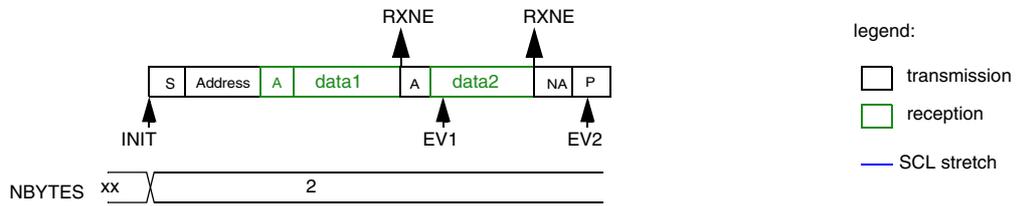


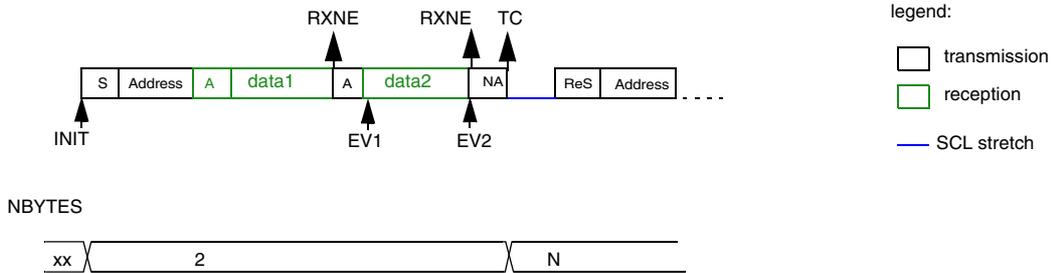
Figure 188. Transfer bus diagrams for I2C master receiver

Example I2C master receiver 2 bytes, automatic end mode (STOP)



INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START  
 EV1: RXNE ISR: rd data1  
 EV2: RXNE ISR: rd data2

Example I2C master receiver 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START  
 EV1: RXNE ISR: rd data1  
 EV2: RXNE ISR: read data2  
 EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19865V1

### 24.4.9 I2Cx\_TIMINGR register configuration examples

The tables below provide examples of how to program the I2Cx\_TIMINGR to obtain timings compliant with the I<sup>2</sup>C specification. In order to get more accurate configuration values, please refer to application note AN4235 *I<sup>2</sup>C timing configuration tool* and the associated software STSW-STM32126.

**Table 72. Examples of timings settings for  $f_{I2CCLK} = 8 \text{ MHz}$**

Parameter	Standard-mode		Fast-mode	Fast-mode Plus
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	1	1	0	0
SCLL	0xC7	0x13	0x9	0x6
$t_{SCLL}$	200x250 ns = 50 $\mu$ s	20x250 ns = 5.0 $\mu$ s	10x125 ns = 1250 ns	7x125 ns = 875 ns
SCLH	0xC3	0xF	0x3	0x3
$t_{SCLH}$	196x250 ns = 49 $\mu$ s	16x250 ns = 4.0 $\mu$ s	4x125ns = 500ns	4x125 ns = 500 ns
$t_{SCL}^{(1)}$	~100 $\mu$ s <sup>(2)</sup>	~10 $\mu$ s <sup>(2)</sup>	~2500 ns <sup>(3)</sup>	~2000 ns <sup>(4)</sup>
SDADEL	0x2	0x2	0x1	0x0
$t_{SDADEL}$	2x250 ns = 500 ns	2x250 ns = 500 ns	1x125 ns = 125 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	5x250 ns = 1250 ns	5x250 ns = 1250 ns	4x125 ns = 500 ns	2x125 ns = 250 ns

1. SCL period  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.
2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}$
3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}$
4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 655 \text{ ns}$

**Table 73. Examples of timings settings for  $f_{I2CCLK} = 16 \text{ MHz}$**

Parameter	Standard-mode		Fast-mode	Fast-mode Plus
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	3	3	1	0
SCLL	0xC7	0x13	0x9	0x4
$t_{SCLL}$	200 x 250 ns = 50 $\mu$ s	20 x 250 ns = 5.0 $\mu$ s	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH	0xC3	0xF	0x3	0x2
$t_{SCLH}$	196 x 250 ns = 49 $\mu$ s	16 x 250 ns = 4.0 $\mu$ s	4 x 125ns = 500 ns	3 x 62.5 ns = 187.5 ns
$t_{SCL}^{(1)}$	~100 $\mu$ s <sup>(2)</sup>	~10 $\mu$ s <sup>(2)</sup>	~2500 ns <sup>(3)</sup>	~1000 ns <sup>(4)</sup>
SDADEL	0x2	0x2	0x2	0x0
$t_{SDADEL}$	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x2
$t_{SCLDEL}$	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. SCL period  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.
2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 1000$  ns
3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 750$  ns
4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 500$  ns

**Table 74. Examples of timings settings for  $f_{I2CCLK} = 48$  MHz**

Parameter	Standard-mode		Fast-mode	Fast-mode Plus
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	5	5
SCLL	0xC7	0x13	0x9	0x3
$t_{SCLL}$	$200 \times 250$ ns = 50 $\mu$ s	$20 \times 250$ ns = 5.0 $\mu$ s	$10 \times 125$ ns = 1250 ns	$4 \times 125$ ns = 500 ns
SCLH	0xC3	0xF	0x3	0x1
$t_{SCLH}$	$196 \times 250$ ns = 49 $\mu$ s	$16 \times 250$ ns = 4.0 $\mu$ s	$4 \times 125$ ns = 500 ns	$2 \times 125$ ns = 250 ns
$t_{SCL}^{(1)}$	$\sim 100$ $\mu$ s <sup>(2)</sup>	$\sim 10$ $\mu$ s <sup>(2)</sup>	$\sim 2500$ ns <sup>(3)</sup>	$\sim 875$ ns <sup>(4)</sup>
SDADEL	0x2	0x2	0x3	0x0
$t_{SDADEL}$	$2 \times 250$ ns = 500 ns	$2 \times 250$ ns = 500 ns	$3 \times 125$ ns = 375 ns	0 ns
SCLDEL	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	$5 \times 250$ ns = 1250 ns	$5 \times 250$ ns = 1250 ns	$4 \times 125$ ns = 500 ns	$2 \times 125$ ns = 250 ns

1. The SCL period  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to the SCL internal detection delay. Values provided for  $t_{SCL}$  are only examples.
2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 1000$  ns
3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 750$  ns
4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 250$  ns

### 24.4.10 SMBus specific features

This section is relevant only when SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

#### Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I<sup>2</sup>C principles of operation. SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification rev 2.0 (<http://smbus.org>).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

## **SMBUS is based on I<sup>2</sup>C specification rev 2.1.**

### **Bus protocols**

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification ver. 2.0 (<http://smbus.org>).

### **Address resolution protocol (ARP)**

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2Cx\_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus Address Resolution Protocol, refer to SMBus specification ver. 2.0 (<http://smbus.org>).

### **Received Command and Data acknowledge control**

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2Cx\_CR1 register. Refer to [Slave Byte Control Mode on page 564](#) section for more details.

### **Host Notify protocol**

This peripheral supports the Host Notify protocol by setting the SMBHEN bit in the I2Cx\_CR1 register. In this case the host will acknowledge the SMBus Host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

### **SMBus alert**

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and

simultaneously accesses all SMBALERT# devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled SMBALERT# low will acknowledge the Alert Response Address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2Cx\_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2Cx\_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

*If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0.*

### Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the  $C(x) = x_8 + x^2 + x + 1$  CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC.

### Timeouts

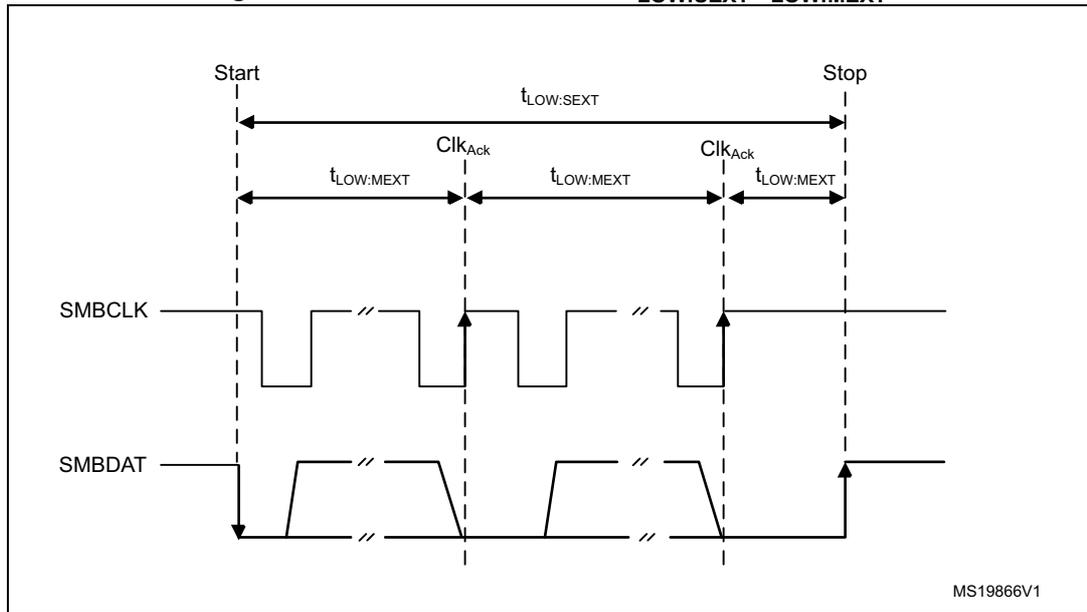
This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification ver. 2.0.

**Table 75. SMBus timeout specifications**

Symbol	Parameter	Limits		Unit
		Min	Max	
t <sub>TIMEOUT</sub>	Detect clock low timeout	25	35	ms
t <sub>LOW:SEXT</sub> <sup>(1)</sup>	Cumulative clock low extend time (slave device)		25	ms
t <sub>LOW:MEXT</sub> <sup>(2)</sup>	Cumulative clock low extend time (master device)		10	ms

1. t<sub>LOW:SEXT</sub> is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master will also extend the clock causing the combined clock low extend time to be greater than t<sub>LOW:SEXT</sub>. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. t<sub>LOW:MEXT</sub> is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master will also extend the clock causing the combined clock low time to be greater than t<sub>LOW:MEXT</sub> on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 189. Timeout intervals for  $t_{LOW:SEXT}$ ,  $t_{LOW:MEXT}$



### Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for  $t_{IDLE}$  greater than  $t_{HIGH,MAX}$ . (refer to [Table 71: I2C-SMBUS specification clock timings](#))

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

### 24.4.11 SMBus initialization

This section is relevant only when SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

#### Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2Cx\_CR1 register. Refer to [Slave Byte Control Mode on page 564](#) for more details.

### Specific address (Slave mode)

The specific SMBus addresses should be enabled if needed. Refer to [Bus idle detection on page 588](#) for more details.

- The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2Cx\_CR1 register.
- The SMBus Host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2Cx\_CR1 register.
- The Alert Response Address (0b0001100) is enabled by setting the ALERTEN bit in the I2Cx\_CR1 register.

### Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2Cx\_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2Cx\_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES-1 data have been transferred when the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

**Caution:** Changing the PECEN configuration is not allowed when the I2C is enabled.

**Table 76. SMBUS with PEC configuration table**

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Master Tx/Rx NBYTES + PEC+ STOP	x	0	1	1
Master Tx/Rx NBYTES + PEC + ReSTART	x	0	0	1
Slave Tx/Rx with PEC	1	0	x	1

### Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits in the I2Cx\_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification ver. 2.0.

- $t_{TIMEOUT}$  check  
 In order to enable the  $t_{TIMEOUT}$  check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value in order to check the  $t_{TIMEOUT}$  parameter. The TIDLE bit must be configured to '0' in order to detect the SCL low level timeout.  
 Then the timer is enabled by setting the TIMOUTEN in the I2Cx\_TIMEOUTR register.  
 If SCL is tied low for a time greater than  $(TIMEOUTA+1) \times 2048 \times t_{I2CCLK}$ , the TIMEOUT flag is set in the I2Cx\_ISR register.  
 Refer to [Table 77: Examples of TIMEOUTA settings for various I2CCLK frequencies \(max  \$t\_{TIMEOUT} = 25\$  ms\)](#).

**Caution:** Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$  and  $t_{LOW:MEXT}$  check  
 Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check  $t_{LOW:SEXT}$  for a slave and  $t_{LOW:MEXT}$  for a master. As the standard specifies only a maximum, you can choose the same value for the both.  
 Then the timer is enabled by setting the TEXTEN bit in the I2Cx\_TIMEOUTR register.  
 If the SMBus peripheral performs a cumulative SCL stretch for a time greater than  $(TIMEOUTB+1) \times 2048 \times t_{I2CCLK}$ , and in the timeout interval described in [Bus idle detection on page 588](#) section, the TIMEOUT flag is set in the I2Cx\_ISR register.  
 Refer to [Table 78: Examples of TIMEOUTB settings for various I2CCLK frequencies](#)

**Caution:** Changing the TIMEOUTB configuration is not allowed when the TEXTEN bit is set.

**Bus Idle detection**

In order to enable the  $t_{IDLE}$  check, the 12-bit TIMEOUTA[11:0] field must be programmed with the timer reload value in order to obtain the  $t_{IDLE}$  parameter. The TIDLE bit must be configured to '1 in order to detect both SCL and SDA high level timeout.

Then the timer is enabled by setting the TIMOUTEN bit in the I2Cx\_TIMEOUTR register.

If both the SCL and SDA lines remain high for a time greater than  $(TIMEOUTA+1) \times 4 \times t_{I2CCLK}$ , the TIMEOUT flag is set in the I2Cx\_ISR register.

Refer to [Table 79: Examples of TIMEOUTA settings for various I2CCLK frequencies \(max tIDLE = 50 μs\)](#)

**Caution:** Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

**24.4.12 SMBus: I2Cx\_TIMEOUTR register configuration examples**

This section is relevant only when SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

- Configuring the maximum duration of  $t_{TIMEOUT}$  to 25 ms:

**Table 77. Examples of TIMEOUTA settings for various I2CCLK frequencies (max  $t_{TIMEOUT} = 25$  ms)**

$f_{I2CCLK}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	$98 \times 2048 \times 125 \text{ ns} = 25 \text{ ms}$
16 MHz	0xC3	0	1	$196 \times 2048 \times 62.5 \text{ ns} = 25 \text{ ms}$
48 MHz	0x249	0	1	$586 \times 2048 \times 20.08 \text{ ns} = 25 \text{ ms}$

- Configuring the maximum duration of  $t_{LOW:SEXT}$  and  $t_{LOW:MEXT}$  to 8 ms:

**Table 78. Examples of TIMEOUTB settings for various I2CCLK frequencies**

$f_{I2CCLK}$	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

- Configuring the maximum duration of  $t_{IDLE}$  to 50  $\mu$ s

**Table 79. Examples of TIMEOUTA settings for various I2CCLK frequencies (max  $t_{IDLE} = 50 \mu$ s)**

$f_{I2CCLK}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIDLE}$
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 $\mu$ s
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 $\mu$ s
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 $\mu$ s

### 24.4.13 SMBus slave mode

This section is relevant only when SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

In addition to I2C slave transfer management (refer to [Section 24.4.7: I2C slave mode](#)) some additional software flowcharts are provided to support SMBus.

#### SMBus Slave transmitter

When the IP is used in SMBus, SBC must be programmed to '1' in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts will be NBYTES-1 and the content of the I2Cx\_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES-1 data transfer.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 190. Transfer sequence flowchart for SMBus slave transmitter N bytes + PEC

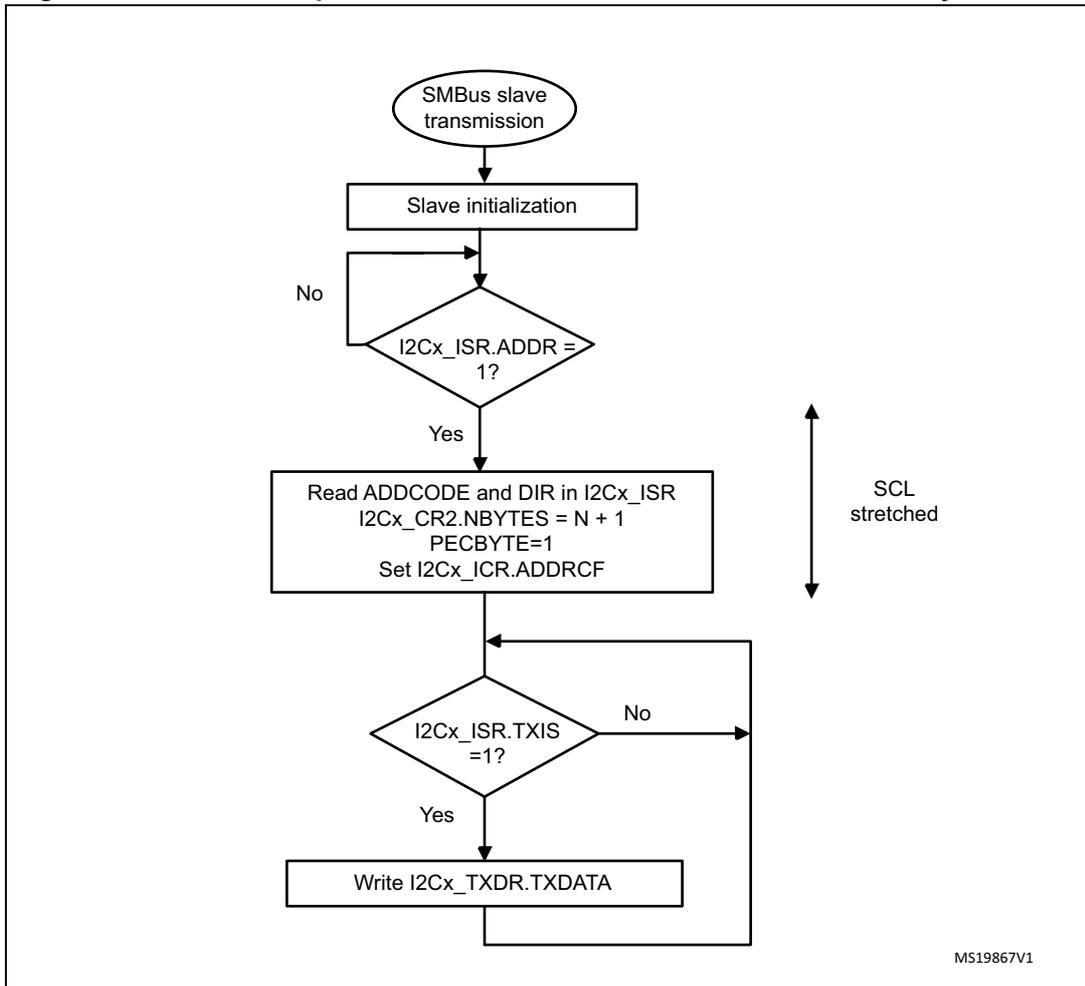
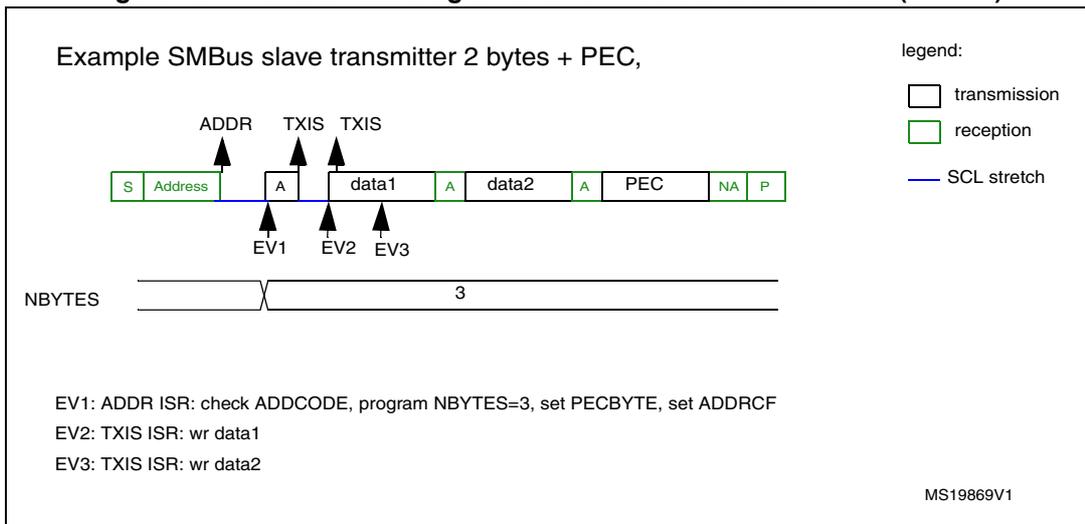


Figure 191. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



### SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to '1' in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to [Slave Byte Control Mode on page 564](#) for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES-1 data have been received, the next received byte is compared with the internal I2Cx\_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2Cx\_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

If no ACK software control is needed, you can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES-1 are received, the next received byte is checked as being the PEC.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 192. Transfer sequence flowchart for SMBus slave receiver N Bytes + PEC

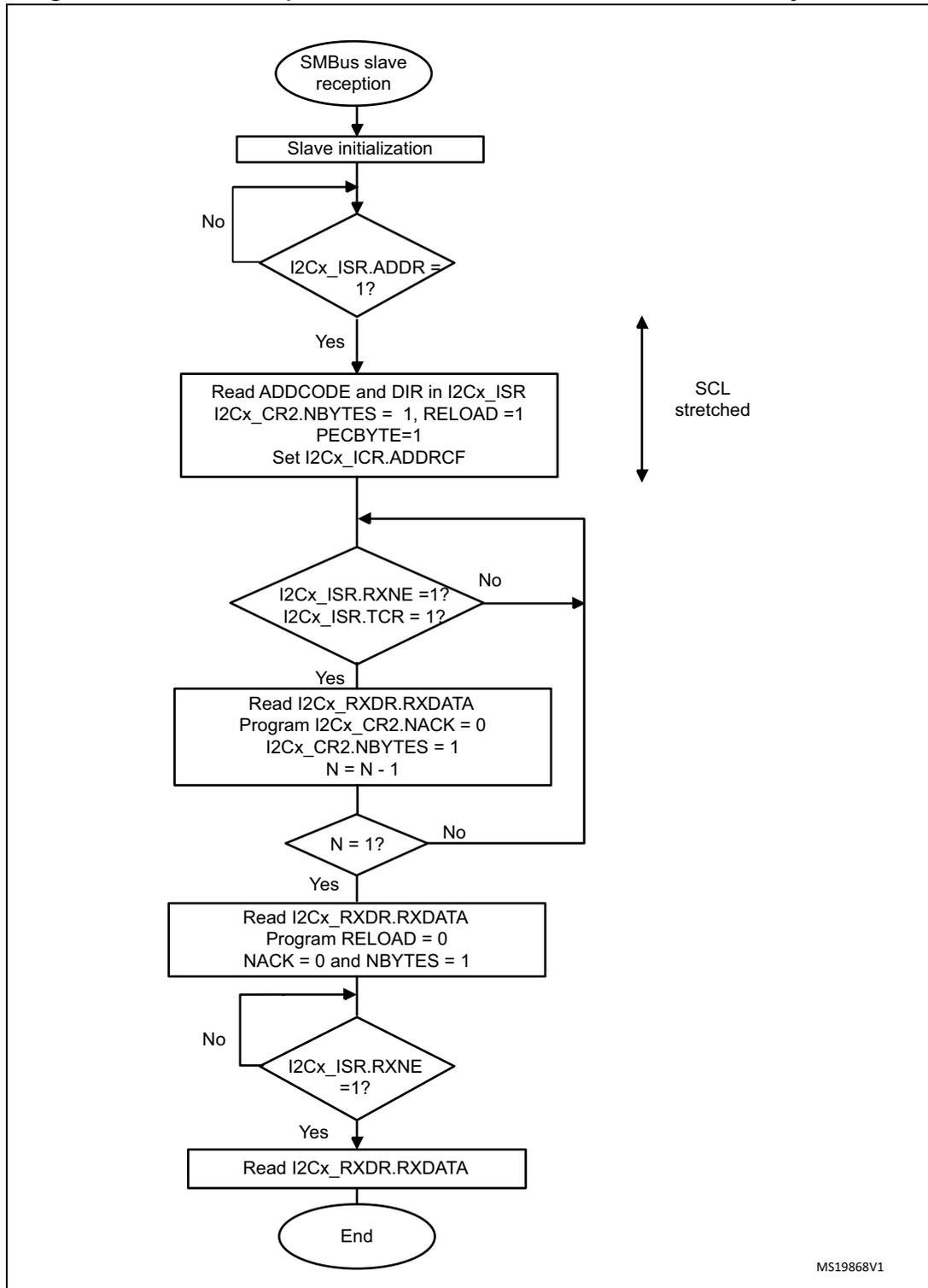
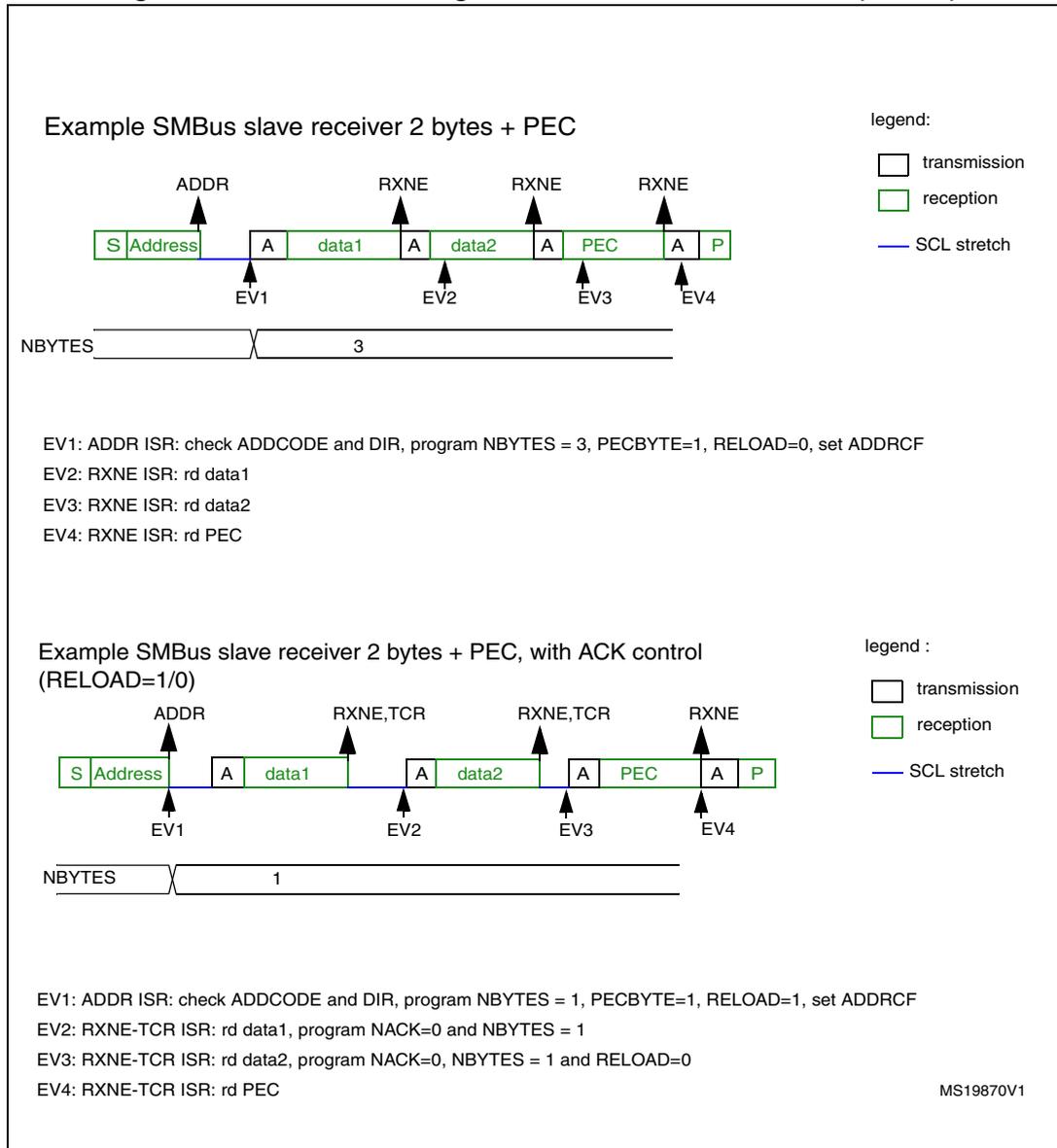


Figure 193. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

In addition to I2C master transfer management (refer to [Section 24.4.8: I2C master mode](#)) some additional software flowcharts are provided to support SMBus.

### SMBus Master transmitter

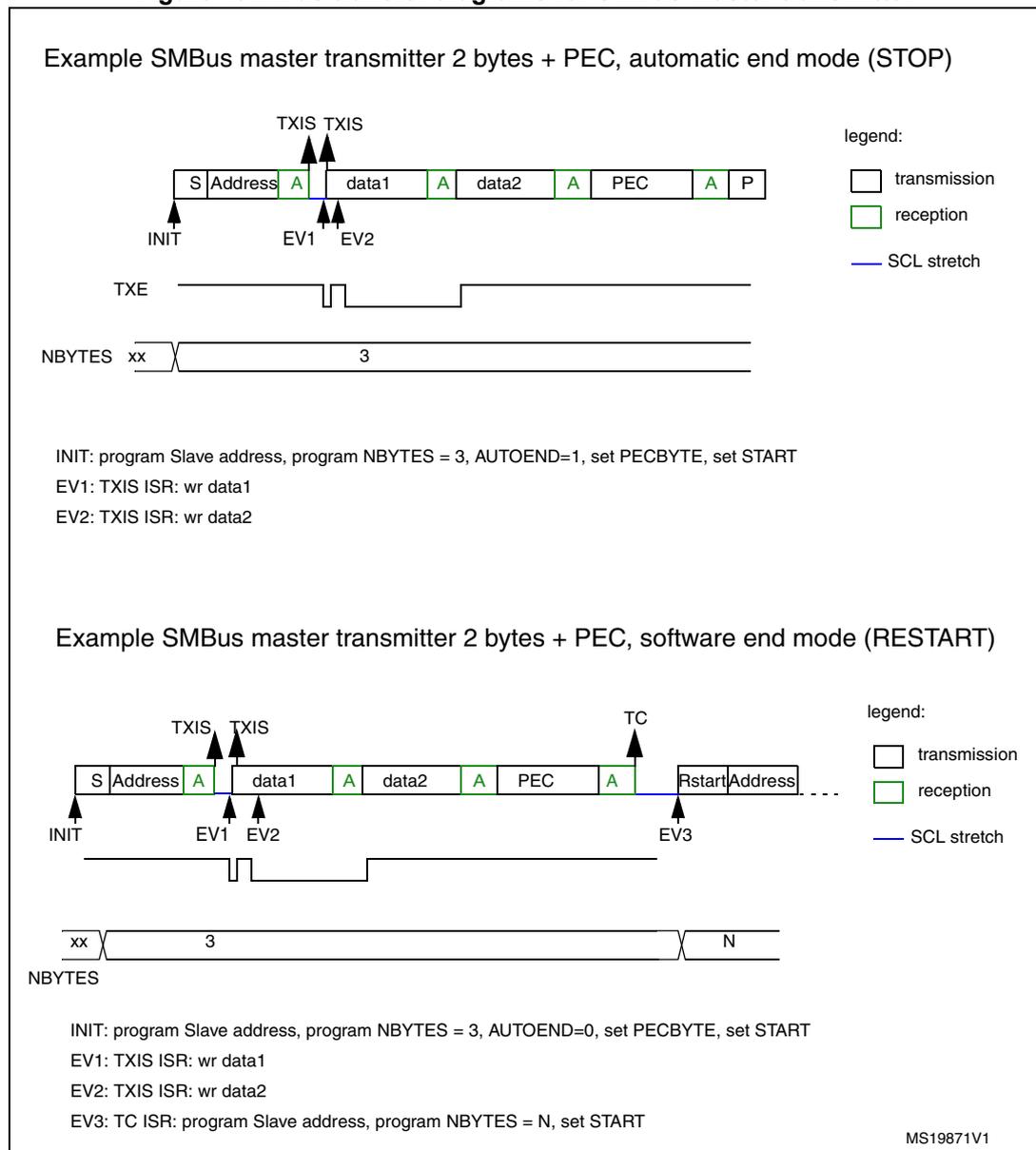
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts will be NBYTES-1. So if the PECBYTE bit is set when NBYTES=0x1, the content of the I2Cx\_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode should be selected (AUTOEND=1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES-1 have been transmitted, the I2Cx\_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 194. Bus transfer diagrams for SMBus master transmitter**



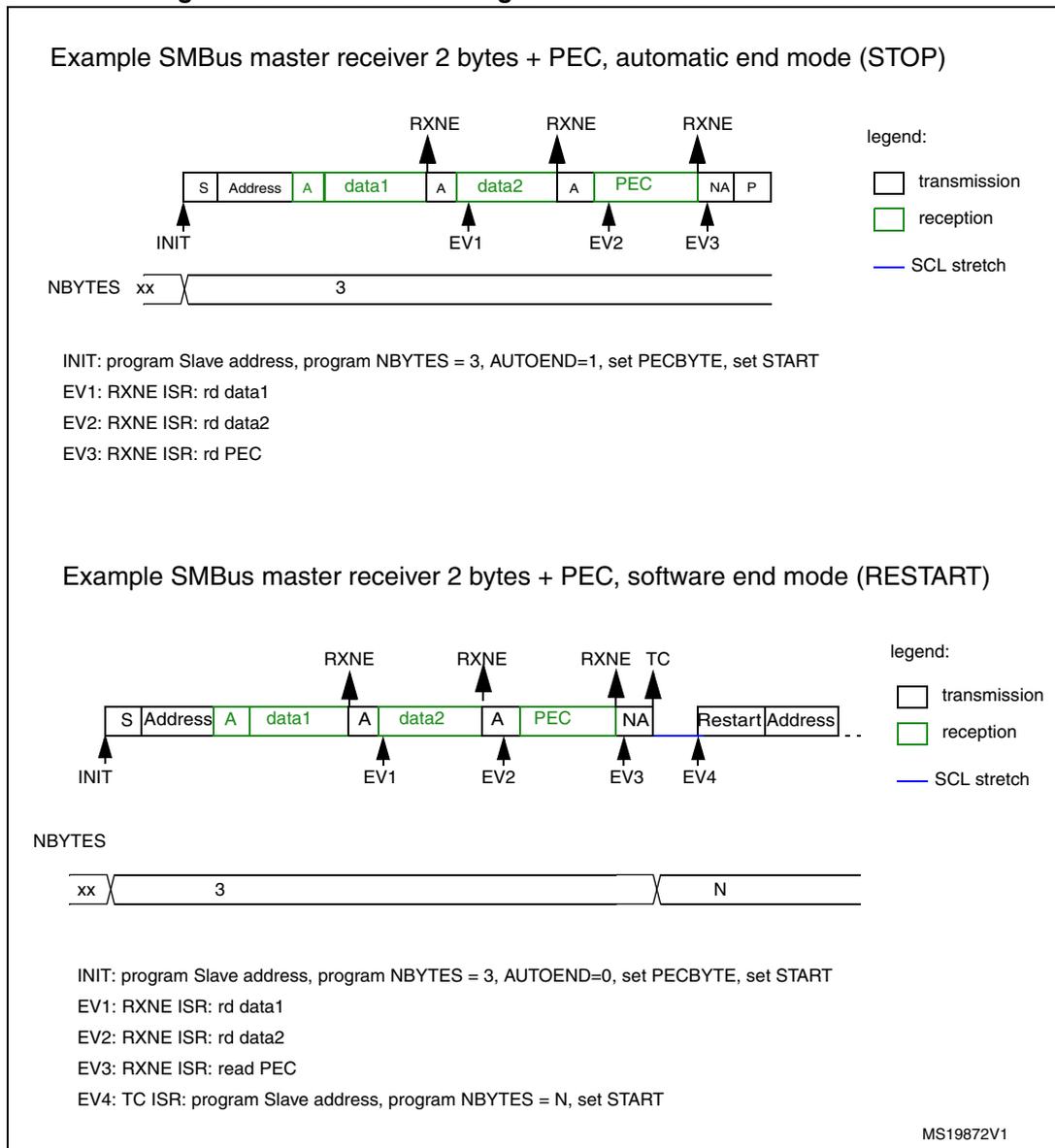
**SMBus Master receiver**

When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND=1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2Cx\_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND=0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES-1 data have been received, the next received byte is automatically checked versus the I2Cx\_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 195. Bus transfer diagrams for SMBus master receiver



### 24.4.14 Wakeup from Stop mode on address match

This section is relevant only when Wakeup from Stop mode feature is supported. Please refer to [Section 24.3: I2C implementation](#).

The I2C is able to wakeup the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wakeup from Stop mode is enabled by setting the WUPEN bit in the I2Cx\_CR1 register. The HSI oscillator must be selected as the clock source for I2CCLK in order to allow wakeup from Stop mode.

During Stop mode, the HSI is switched off. When a START is detected, the I2C interface switches the HSI on, and stretches SCL low until HSI is woken up.

HSI is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wakeup time. The stretch is released when ADDR flag is cleared by software, and the transfer goes on normally.

If the address does not match, the HSI is switched off again and the MCU is not woken up.

**Note:** *If the I2C clock is the system clock, or if WUPEN = 0, the HSI oscillator is not switched on after a START is received.*

*Only an ADDR interrupt can wakeup the MCU. Therefore do not enter Stop mode when the I2C is performing a transfer as a master, or as an addressed slave after the ADDR flag is set. This can be managed by clearing SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.*

**Caution:** The digital filter is not compatible with the wakeup from Stop mode feature. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

**Caution:** This feature is available only when the I2C clock source is the HSI oscillator.

**Caution:** Clock stretching must be enabled (NOSTRETCH=0) to ensure proper operation of the wakeup from Stop mode feature.

**Caution:** If wakeup from Stop mode is disabled (WUPEN=0), the I2C peripheral must be disabled before entering Stop mode (PE=0).

#### 24.4.15 Error conditions

The following are the error conditions which may cause communication to fail.

##### Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of 9 SCL clock pulses. A START or a STOP condition is detected when a SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2Cx\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

##### Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2Cx\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

### Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH=1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF=1 and the first data byte should be sent. The content of the I2Cx\_TXDR register is sent if TXE=0, 0xFF if not.
  - When a new byte should be sent and the I2Cx\_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2Cx\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

### Packet Error Checking Error (PECERR)

This section is relevant only when the SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

A PEC error is detected when the received PEC byte does not match with the I2Cx\_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2Cx\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

### Timeout Error (TIMEOUT)

This section is relevant only when the SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

A timeout error occurs for any of these conditions:

- TIDLE=0 and SCL remained low for the time defined in the TIMEOUTA[11:0] bits: this is used to detect a SMBus timeout.
- TIDLE=1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus  $t_{LOW:MEXT}$  parameter)
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus  $t_{LOW:SEXT}$  parameter)

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2Cx\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

### Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Please refer to [Section 24.3: I2C implementation](#).

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2Cx\_CR1 register.

## 24.4.16 DMA requests

### Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2Cx\_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see [Section 10: Direct memory access controller \(DMA\) on page 162](#)) to the I2Cx\_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to [Master transmitter on page 576](#).
- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to [SMBus Slave transmitter on page 591](#) and [SMBus Master transmitter on page 595](#).

*Note:* If DMA is used for transmission, the TXIE bit does not need to be enabled.

### Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2Cx\_CR1 register. Data is loaded from the I2Cx\_RXDR register to an SRAM area configured using the DMA peripheral (refer to [Section 10: Direct memory access controller \(DMA\) on page 162](#)) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see [Section 24.3: I2C implementation](#)): the PEC transfer is managed with the NBYTES counter. Refer to [SMBus Slave receiver on page 593](#) and [SMBus Master receiver on page 597](#).

*Note:* If DMA is used for reception, the RXIE bit does not need to be enabled.

### 24.4.17 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG\_I2Cx\_SMBUS\_TIMEOUT configuration bits in the DBG module.

## 24.5 I2C low-power modes

Table 80. low-power modes

Mode	Description
Sleep	No effect I2C interrupts cause the device to exit the Sleep mode.
Stop	The I2C registers content is kept. If WUPEN=1: the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. If WUPEN=0: the I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby.

## 24.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 81. I2C Interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Receive buffer not empty	RXNE	Read I2Cx_RXDR register	RXIE
Transmit buffer interrupt status	TXIS	Write I2Cx_TXDR register	TXIE
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE
Transfer Complete Reload	TCR	Write I2Cx_CR2 with NBYTES[7:0] ≠ 0	TCIE
Transfer complete	TC	Write START=1 or STOP=1	
Address matched	ADDR	Write ADDRCF=1	ADDRIE
NACK reception	NACKF	Write NACKCF=1	NACKIE

Table 81. I2C Interrupt requests (continued)

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Bus error	BERR	Write BERRCF=1	ERRIE
Arbitration loss	ARLO	Write ARLOCF=1	
Overrun/Underrun	OVR	Write OVRCF=1	
PEC error	PECERR	Write PECERRCF=1	
Timeout/t <sub>LOW</sub> error	TIMEOUT	Write TIMEOUTCF=1	
SMBus Alert	ALERT	Write ALERTCF=1	

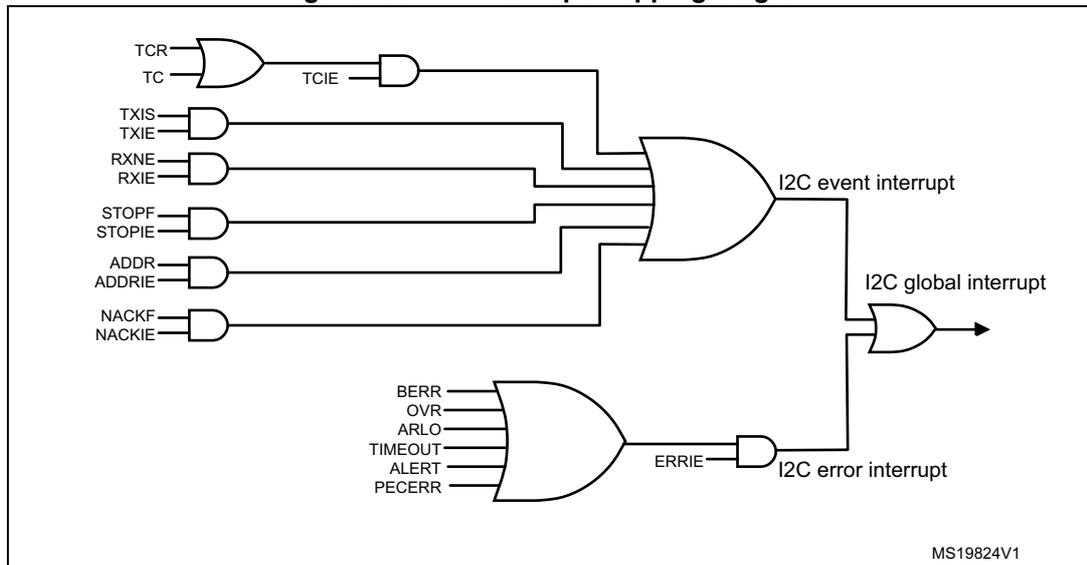
Depending on the product implementation, all these interrupts events can either share the same interrupt vector (I2C global interrupt), or be grouped into 2 interrupt vectors (I2C event interrupt and I2C error interrupt). Refer to [Table 28: List of vectors](#) for details.

In order to enable the I2C interrupts, the following sequence is required:

1. Configure and enable the I2C IRQ channel in the NVIC.
2. Configure the I2C to generate interrupts.

The I2C wakeup event is connected to the EXTI controller (refer to [Section 11.2: Extended interrupts and events controller \(EXTI\)](#)).

Figure 196. I2C interrupt mapping diagram



## 24.7 I2C registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

### 24.7.1 Control register 1 (I2Cx\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBDEN	SMBHEN	GCEN	WUPE N	NOSTR ETCH	SBC	
								rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RXDMA EN	TXDMA EN	Res.	ANF OFF	DNF				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE	
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 **PECEN**: PEC enable

- 0: PEC calculation disabled
- 1: PEC calculation enabled

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 24.3: I2C implementation](#).*

Bit 22 **ALERTEN**: SMBus alert enable

**Device mode (SMBHEN=0):**

- 0: Releases SMBA pin high and Alert Response Address Header disabled: 0001100x followed by NACK.
- 1: Drives SMBA pin low and Alert Response Address Header enables: 0001100x followed by ACK.

**Host mode (SMBHEN=1):**

- 0: SMBus Alert pin (SMBA) not supported.
- 1: SMBus Alert pin (SMBA) supported.

*Note: When ALERTEN=0, the SMBA pin can be used as a standard GPIO.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 24.3: I2C implementation](#).*

Bit 21 **SMBDEN**: SMBus Device Default address enable

- 0: Device default address disabled. Address 0b1100001x is NACKed.
- 1: Device default address enabled. Address 0b1100001x is ACKed.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 24.3: I2C implementation](#).*

Bit 20 **SMBHEN**: SMBus Host address enable

- 0: Host address disabled. Address 0b0001000x is NACKed.
- 1: Host address enabled. Address 0b0001000x is ACKed.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 24.3: I2C implementation](#).*

- Bit 19 **GCEN**: General call enable  
 0: General call disabled. Address 0b00000000 is NACKed.  
 1: General call enabled. Address 0b00000000 is ACKed.
- Bit 18 **WUPEN**: Wakeup from Stop mode enable  
 0: Wakeup from Stop mode disable.  
 1: Wakeup from Stop mode enable.  
*Note: If the Wakeup from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 24.3: I2C implementation](#).*  
*Note: WUPEN can be set only when DNF = '0000'*
- Bit 17 **NOSTRETCH**: Clock stretching disable  
 This bit is used to disable clock stretching in slave mode.  
 0: Clock stretching enabled  
 1: Clock stretching disabled  
*Note: This bit can only be programmed when the I2C is disabled (PE = 0).*
- Bit 16 **SBC**: Slave byte control  
 This bit is used to enable hardware byte control in slave mode.  
 0: Slave byte control disabled  
 1: Slave byte control enabled
- Bit 15 **RXDMAEN**: DMA reception requests enable  
 0: DMA mode disabled for reception  
 1: DMA mode enabled for reception
- Bit 14 **TXDMAEN**: DMA transmission requests enable  
 0: DMA mode disabled for transmission  
 1: DMA mode enabled for transmission
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **ANFOFF**: Analog noise filter OFF  
 0: Analog noise filter enabled  
 1: Analog noise filter disabled  
*Note: This bit can only be programmed when the I2C is disabled (PE = 0).*
- Bits 11:8 **DNF[3:0]**: Digital noise filter  
 These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter will filter spikes with a length of up to  $DNF[3:0] * t_{I2CCLK}$   
 0000: Digital filter disabled  
 0001: Digital filter enabled and filtering capability up to  $1 t_{I2CCLK}$   
 ...  
 1111: digital filter enabled and filtering capability up to  $15 t_{I2CCLK}$   
*Note: If the analog filter is also enabled, the digital filter is added to the analog filter.  
 This filter can only be programmed when the I2C is disabled (PE = 0).*

Bit 7 **ERRIE**: Error interrupts enable

0: Error detection interrupts disabled

1: Error detection interrupts enabled

*Note: Any of these errors generate an interrupt:*

*Arbitration Loss (ARLO)*

*Bus Error detection (BERR)*

*Overrun/Underrun (OVR)*

*Timeout detection (TIMEOUT)*

*PEC error detection (PECERR)*

*Alert pin event detection (ALERT)*

Bit 6 **TCIE**: Transfer Complete interrupt enable

0: Transfer Complete interrupt disabled

1: Transfer Complete interrupt enabled

*Note: Any of these events will generate an interrupt:*

*Transfer Complete (TC)*

*Transfer Complete Reload (TCR)*

Bit 5 **STOPIE**: STOP detection Interrupt enable

0: Stop detection (STOPF) interrupt disabled

1: Stop detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received Interrupt enable

0: Not acknowledge (NACKF) received interrupts disabled

1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match Interrupt enable (slave only)

0: Address match (ADDR) interrupts disabled

1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX Interrupt enable

0: Receive (RXNE) interrupt disabled

1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX Interrupt enable

0: Transmit (TXIS) interrupt disabled

1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable

0: Peripheral disable

1: Peripheral enable

*Note: When PE=0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least 3 APB clock cycles.*

### 24.7.2 Control register 2 (I2Cx\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD 10R	ADD10	RD_W RN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw									

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

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address Matched is received, also when PE=0.

0: No PEC transfer.

1: PEC transmission/reception is requested

*Note: Writing '0' to this bit has no effect.*

*This bit has no effect when RELOAD is set.*

*This bit has no effect is slave mode when SBC=0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'.*

*Please refer to [Section 24.3: I2C implementation](#).*

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

*Note: This bit has no effect in slave mode or when the RELOAD bit is set.*

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART will follow).

1: The transfer is not completed after the NBYTES data transfer (NBYTES will be reloaded).

TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

*Note: Changing these bits when the START bit is set is not allowed.*

**Bit 15 NACK:** NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address Matched is received, or when PE=0.

- 0: an ACK is sent after current received byte.
- 1: a NACK is sent after current received byte.

*Note: Writing '0' to this bit has no effect.*

*This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.*

*When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.*

*When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.*

**Bit 14 STOP:** Stop generation (master mode)

The bit is set by software, cleared by hardware when a Stop condition is detected, or when PE = 0.

**In Master Mode:**

- 0: No Stop generation.
- 1: Stop generation after current byte transfer.

*Note: Writing '0' to this bit has no effect.*

**Bit 13 START:** Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software by writing '1' to the ADDRCONF bit in the I2Cx\_ICR register.

- 0: No Start generation.
- 1: Restart/Start generation:

- If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.
- Otherwise setting this bit will generate a START condition once the bus is free.

*Note: Writing '0' to this bit has no effect.*

*The START bit can be set even if the bus is BUSY or I2C is in slave mode.*

*This bit has no effect when RELOAD is set.*

**Bit 12 HEAD10R:** 10-bit address header only read direction (master receiver mode)

- 0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.
- 1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

*Note: Changing this bit when the START bit is set is not allowed.*

**Bit 11 ADD10:** 10-bit addressing mode (master mode)

- 0: The master operates in 7-bit addressing mode,
- 1: The master operates in 10-bit addressing mode

*Note: Changing this bit when the START bit is set is not allowed.*

**Bit 10 RD\_WRN:** Transfer direction (master mode)

- 0: Master requests a write transfer.
- 1: Master requests a read transfer.

*Note: Changing this bit when the START bit is set is not allowed.*

Bits 9:8 **SADD[9:8]**: Slave address bit 9:8 (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

These bits are don't care

**In 10-bit addressing mode (ADD10 = 1):**

These bits should be written with bits 9:8 of the slave address to be sent

*Note: Changing these bits when the START bit is set is not allowed.*

Bits 7:1 **SADD[7:1]**: Slave address bit 7:1 (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

These bits should be written with the 7-bit slave address to be sent

**In 10-bit addressing mode (ADD10 = 1):**

These bits should be written with bits 7:1 of the slave address to be sent.

*Note: Changing these bits when the START bit is set is not allowed.*

Bit 0 **SADD0**: Slave address bit 0 (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

This bit is don't care

**In 10-bit addressing mode (ADD10 = 1):**

This bit should be written with bit 0 of the slave address to be sent

*Note: Changing these bits when the START bit is set is not allowed.*

### 24.7.3 Own address 1 register (I2Cx\_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1 MODE	OA1[9:8]		OA1[7:1]							OA1[0]
rw					rw	rw		rw							rw

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

Bit 15 **OA1EN**: Own Address 1 enable

- 0: Own address 1 disabled. The received slave address OA1 is NACKed.
- 1: Own address 1 enabled. The received slave address OA1 is ACKed.

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

Bit 10 **OA1MODE** Own Address 1 10-bit mode

- 0: Own address 1 is a 7-bit address.
- 1: Own address 1 is a 10-bit address.

*Note: This bit can be written only when OA1EN=0.*

Bits 9:8 **OA1[9:8]**: Interface address

- 7-bit addressing mode: don't care
- 10-bit addressing mode: bits 9:8 of address

*Note: These bits can be written only when OA1EN=0.*

Bits 7:1 **OA1[7:1]**: Interface address

bits 7:1 of address

*Note: These bits can be written only when OA1EN=0.*

Bit 0 **OA1[0]**: Interface address

- 7-bit addressing mode: don't care
- 10-bit addressing mode: bit 0 of address

*Note: This bit can be written only when OA1EN=0.*

### 24.7.4 Own address 2 register (I2Cx\_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw			rw							

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

Bit 15 **OA2EN**: Own Address 2 enable

- 0: Own address 2 disabled. The received slave address OA2 is NACKed.
- 1: Own address 2 enabled. The received slave address OA2 is ACKed.

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

Bits 10:8 **OA2MSK[2:0]**: Own Address 2 masks

- 000: No mask
- 001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
- 010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
- 011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
- 100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
- 101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
- 110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
- 111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

*Note: These bits can be written only when OA2EN=0.*

*As soon as OA2MSK is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if the comparison matches.*

Bits 7:1 **OA2[7:1]**: Interface address

bits 7:1 of address

*Note: These bits can be written only when OA2EN=0.*

Bit 0 Reserved, must be kept at reset value.

### 24.7.5 Timing register (I2Cx\_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait states

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

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK in order to generate the clock period  $t_{PRESC}$  used for data setup and hold counters (refer to [I2C timings on page 557](#)) and for SCL high and low level counters (refer to [I2C master initialization on page 572](#)).

$$t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$$

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay  $t_{SCLDEL}$  between SDA edge and SCL rising edge in transmission mode.

$$t_{SCLDEL} = (SCLDEL+1) \times t_{PRESC}$$

Note:  $t_{SCLDEL}$  is used to generate  $t_{SU:DAT}$  timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay  $t_{SDADEL}$  between SCL falling edge SDA edge in transmission mode.

$$t_{SDADEL} = SDADEL \times t_{PRESC}$$

Note:  $SDADEL$  is used to generate  $t_{HD:DAT}$  timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

$$t_{SCLH} = (SCLH+1) \times t_{PRESC}$$

Note:  $SCLH$  is also used to generate  $t_{SU:STO}$  and  $t_{HD:STA}$  timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

$$t_{SCLL} = (SCLL+1) \times t_{PRESC}$$

Note:  $SCLL$  is also used to generate  $t_{BUF}$  and  $t_{SU:STA}$  timings.

Note: This register must be configured when the I2C is disabled ( $PE = 0$ ).

### 24.7.6 Timeout register (I2Cx\_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times PCLK1 + 6 \times I2CCLK$ .

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

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than  $t_{LOW:EXT}$  is done by the I2C interface, a timeout error is detected (TIMEOUT=1).

Bits 30:29 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

In master mode, the master cumulative clock low extend time ( $t_{LOW:MEXT}$ ) is detected

In slave mode, the slave cumulative clock low extend time ( $t_{LOW:SEXT}$ ) is detected

$$t_{LOW:EXT} = (TIMEOUTB + 1) \times 2048 \times t_{I2CCLK}$$

Note: These bits can be written only when TEXTEN=0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled: when SCL is low for more than  $t_{TIMEOUT}$  (TIDLE=0) or high for more than  $t_{IDLE}$  (TIDLE=1), a timeout error is detected (TIMEOUT=1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN=0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A

This field is used to configure:

– The SCL low timeout condition  $t_{TIMEOUT}$  when TIDLE=0

$$t_{TIMEOUT} = (TIMEOUTA + 1) \times 2048 \times t_{I2CCLK}$$

– The bus idle condition (both SCL and SDA high) when TIDLE=1

$$t_{IDLE} = (TIMEOUTA + 1) \times 4 \times t_{I2CCLK}$$

Note: These bits can be written only when TIMOUTEN=0.

Note: If the SMBus feature is not supported, this register is reserved and forced by hardware to “0x00000000”. Please refer to [Section 24.3: I2C implementation](#).

### 24.7.7 Interrupt and Status register (I2Cx\_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]						DIR	
								r						r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIME OUT	PEC ERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	r_w1	r_w1

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

Bits 23:17 **ADDCODE[6:0]**: Address match code (Slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the 2 MSBs of the address.

Bit 16 **DIR**: Transfer direction (Slave mode)

This flag is updated when an address match event occurs (ADDR=1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a Stop condition is detected, or when PE=0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN=1 (SMBus host configuration), ALERTEN=1 and a SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

*Note: This bit is cleared by hardware when PE=0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 24.3: I2C implementation](#).*

Bit 12 **TIMEOUT**: Timeout or t<sub>LOW</sub> detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

*Note: This bit is cleared by hardware when PE=0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 24.3: I2C implementation](#).*

- Bit 11 **PECERR**: PEC Error in reception  
This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.  
*Note: This bit is cleared by hardware when PE=0.  
If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'.  
Please refer to [Section 24.3: I2C implementation](#).*
- Bit 10 **OVR**: Overrun/Underrun (slave mode)  
This flag is set by hardware in slave mode with NOSTRETCH=1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 9 **ARLO**: Arbitration lost  
This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 8 **BERR**: Bus error  
This flag is set by hardware when a misplaced Start or Stop condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting *BERRCF bit*.  
*Note: This bit is cleared by hardware when PE=0.*
- Bits 7 **TCR**: Transfer Complete Reload  
This flag is set by hardware when RELOAD=1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.  
*Note: This bit is cleared by hardware when PE=0.  
This flag is only for master mode, or for slave mode when the SBC bit is set.*
- Bit 6 **TC**: Transfer Complete (master mode)  
This flag is set by hardware when RELOAD=0, AUTOEND=0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 5 **STOPF**: Stop detection flag  
This flag is set by hardware when a Stop condition is detected on the bus and the peripheral is involved in this transfer:  
– either as a master, provided that the STOP condition is generated by the peripheral.  
– or as a slave, provided that the peripheral has been addressed previously during this transfer.  
It is cleared by software by setting the STOPCF bit.  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 4 **NACKF**: Not Acknowledge received flag  
This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 3 **ADDR**: Address matched (slave mode)  
This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting *ADDRCF bit*.  
*Note: This bit is cleared by hardware when PE=0.*

- Bit 2 **RXNE**: Receive data register not empty (receivers)  
 This bit is set by hardware when the received data is copied into the I2Cx\_RXDR register, and is ready to be read. It is cleared when I2Cx\_RXDR is read.  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 1 **TXIS**: Transmit interrupt status (transmitters)  
 This bit is set by hardware when the I2Cx\_TXDR register is empty and the data to be transmitted must be written in the I2Cx\_TXDR register. It is cleared when the next data to be sent is written in the I2Cx\_TXDR register.  
 This bit can be written to '1' by software when NOSTRETCH=1 only, in order to generate a TXIS event (interrupt if TXIE=1 or DMA request if TXDMAEN=1).  
*Note: This bit is cleared by hardware when PE=0.*
- Bit 0 **TXE**: Transmit data register empty (transmitters)  
 This bit is set by hardware when the I2Cx\_TXDR register is empty. It is cleared when the next data to be sent is written in the I2Cx\_TXDR register.  
 This bit can be written to '1' by software in order to flush the transmit data register I2Cx\_TXDR.  
*Note: This bit is set by hardware when PE=0.*

### 24.7.8 Interrupt clear register (I2Cx\_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERT CF	TIM OUTCF	PECCF	OVRCF	ARLO CF	BERR CF	Res.	Res.	STOP CF	NACK CF	ADDR CF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

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

- Bit 13 **ALERTCF**: Alert flag clear  
 Writing 1 to this bit clears the ALERT flag in the I2Cx\_ISR register.  
*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to Section 24.3: I2C implementation.*
- Bit 12 **TIMOUTCF**: Timeout detection flag clear  
 Writing 1 to this bit clears the TIMEOUT flag in the I2Cx\_ISR register.  
*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to Section 24.3: I2C implementation.*
- Bit 11 **PECCF**: PEC Error flag clear  
 Writing 1 to this bit clears the PECERR flag in the I2Cx\_ISR register.  
*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to Section 24.3: I2C implementation.*



- Bit 10 **OVRCF**: Overrun/Underrun flag clear  
Writing 1 to this bit clears the OVR flag in the I2Cx\_ISR register.
- Bit 9 **ARLOCF**: Arbitration Lost flag clear  
Writing 1 to this bit clears the ARLO flag in the I2Cx\_ISR register.
- Bit 8 **BERRCF**: Bus error flag clear  
Writing 1 to this bit clears the BERRF flag in the I2Cx\_ISR register.
- Bits 7:6 Reserved, must be kept at reset value.
- Bit 5 **STOPCF**: Stop detection flag clear  
Writing 1 to this bit clears the STOPF flag in the I2Cx\_ISR register.
- Bit 4 **NACKCF**: Not Acknowledge flag clear  
Writing 1 to this bit clears the ACKF flag in I2Cx\_ISR register.
- Bit 3 **ADDRCF**: Address Matched flag clear  
Writing 1 to this bit clears the ADDR flag in the I2Cx\_ISR register. Writing 1 to this bit also clears the START bit in the I2Cx\_CR2 register.
- Bits 2:0 Reserved, must be kept at reset value.

### 24.7.9 PEC register (I2Cx\_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: No wait states

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

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

- Bits 7:0 **PEC[7:0]** Packet error checking register  
This field contains the internal PEC when PECEN=1.  
The PEC is cleared by hardware when PE=0.

*Note:* If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Please refer to [Section 24.3: I2C implementation](#).

**24.7.10 Receive data register (I2Cx\_RXDR)**

Address offset: 0x24

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **RXDATA[7:0]** 8-bit receive data

Data byte received from the I<sup>2</sup>C bus.

**24.7.11 Transmit data register (I2Cx\_TXDR)**

Address offset: 0x28

Reset value: 0x0000 0000

Access: No wait states

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

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

Bits 7:0 **TXDATA[7:0]** 8-bit transmit data

Data byte to be transmitted to the I<sup>2</sup>C bus.

*Note: These bits can be written only when TXE=1.*

24.7.12 I2C register map

The table below provides the I2C register map and reset values.

Table 82. 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
0x0	I2Cx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]			ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE	
	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	
0x4	I2Cx_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]										
	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
0x8	I2Cx_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xC	I2Cx_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]	OA2[7:1]					Res.				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	I2Cx_TIMINGR	PRESC[3:0]			Res.	Res.	Res.	Res.	Res.	SCLDEL[3:0]	SDADEL[3:0]	SCLH[7:0]					SCLL[7:0]																
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2Cx_TIMEOUTR	TEXTEN	Res.	Res.	Res.	Res.	TIMEOUTB[11:0]							Res.	Res.	Res.	TIDLE	TIMEOUTA[11:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2Cx_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDRF	RXNE	TXIS	TXE
	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
0x1C	I2Cx_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2Cx_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x24	I2Cx_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																



Table 82. I2C register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
0x28	I2Cx_TXDR	Res.	TXDATA[7:0]																																				
	Reset value																										0	0	0	0	0	0	0	0					

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 25 Universal synchronous asynchronous receiver transmitter (USART)

### 25.1 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 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.

### 25.2 USART main features

- Full duplex, asynchronous communications
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or by 8 to give flexibility between speed and clock tolerance
- Baud rate generator systems
  - A common programmable transmit and receive baud rate of up to 9 Mbit/s when the clock frequency is 72 MHz and oversampling is by 8
- Dual clock domain allowing
  - UART functionality and wakeup from Stop mode
  - Convenient baud rate programming independent from the PCLK reprogramming
- Auto baud rate detection
- Programmable data word length (8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits - support for 1 or 2 stop bits
- Synchronous mode and clock output for synchronous communication
- Single-wire half-duplex communication
- Continuous communication using DMA (direct memory access)
  - Buffering of received/transmitted bytes in reserved SRAM using centralized DMA
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Fourteen 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
  - Address/character match
  - Receiver timeout interrupt
  - End of block interrupt
  - Wakeup from Stop mode
- Multiprocessor communication - enter mute mode if address match does not occur
- Wakeup from mute mode (by idle line detection or address mark detection)
- Two receiver wakeup modes: Address bit (MSB, 9<sup>th</sup> bit), Idle line

### 25.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder
  - Support for 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T=0 and T=1 asynchronous protocols for Smartcards as defined in the ISO/IEC 7816-3 standard
  - 1.5 stop bits for Smartcard operation
- Support for ModBus communication
  - Timeout feature
  - CR/LF character recognition

## 25.4 USART implementation

This manual describes the full set of features implemented in USART1. In the STM32F37xx devices, USART1, USART2 and USART3 are identical and implement the full set of features as shown in the following table.

**Table 83. USART features**

USART modes/features <sup>(1)</sup>	USART1	USART2	USART3
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode	X	X	X
Smartcard mode	X	X	X
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	X
LIN mode	X	X	X
Dual clock domain and wakeup from Stop mode	X	X	X
Receiver timeout interrupt	X	X	X
Modbus communication	X	X	X
Auto baud rate detection	X	X	X
Driver Enable	X	X	X

1. X = supported.

## 25.5 USART functional description

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 I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire and Smartcard modes, this I/O is used to transmit and receive the data.

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
- 1, 1.5, 2 stop bits indicating that the frame is complete
- This interface uses a baud rate generator
- A status register (USART\_ISR)
- Receive and transmit data registers (USART\_RDR, USART\_TDR)
- A baud rate register (USART\_BRR)
- A guardtime register (USART\_GTPR) in case of Smartcard mode.

Refer to [Section 25.7: USART registers on page 662](#) for the definitions of each bit.

The following pin is required to interface in synchronous mode and Smartcard mode:

- **SCLK**: 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, SCLK output can provide the clock to the Smartcard.

The following pins are required in Hardware flow control mode:

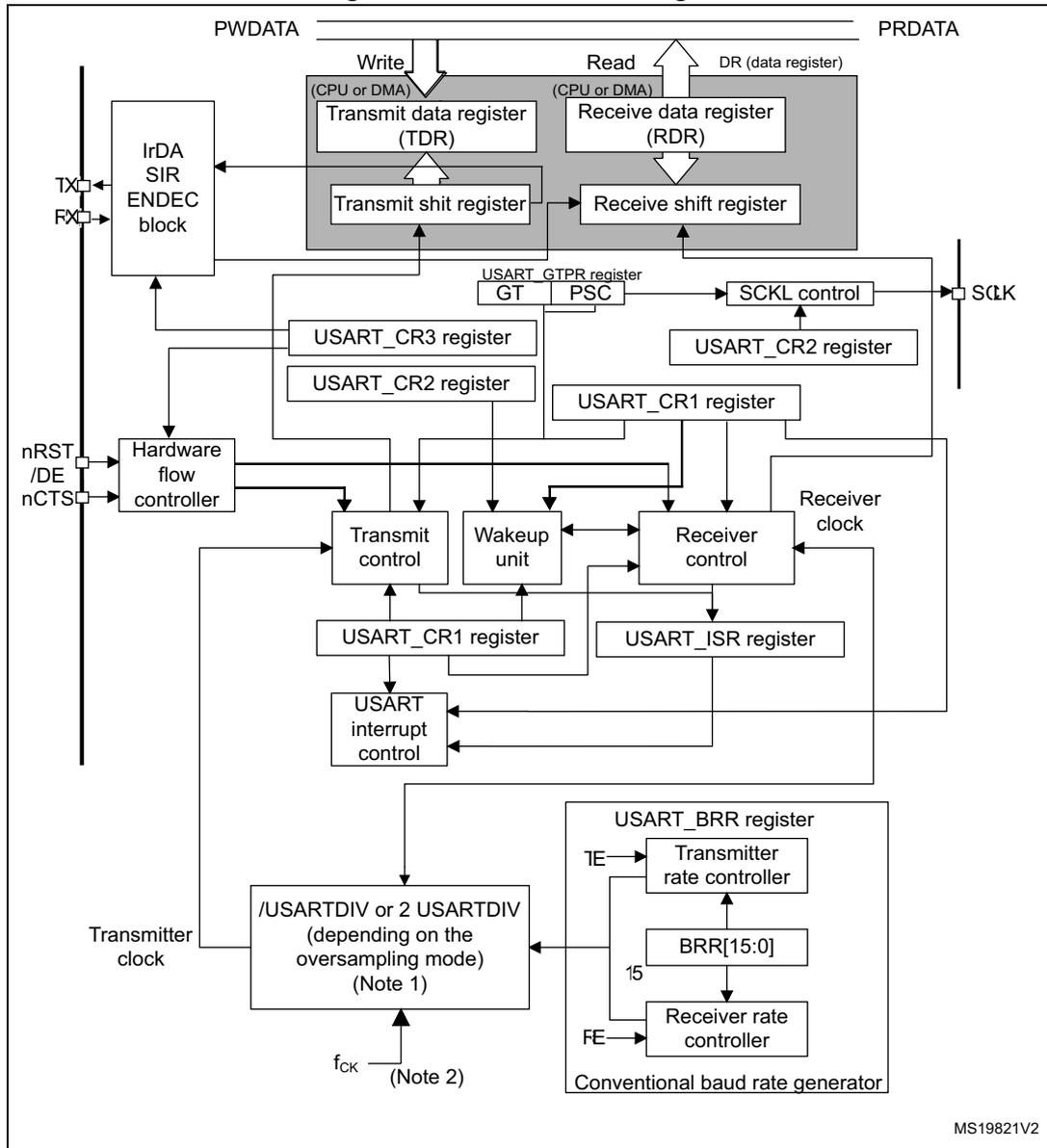
- **nCTS**: Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS**: Request to send indicates that the USART is ready to receive data (when low).

The following pin is required in RS485 Hardware control mode:

- **DE**: Driver Enable activates the transmission mode of the external transceiver.

*Note:* **DE** and **nRTS** share the same pin.

Figure 197. USART block diagram



MS19821V2

1. For details about USARTDIV coding in the USART\_BRR register, please refer to [Section 25.5.4: Baud rate generation](#).
2. f<sub>CK</sub> can be f<sub>LSE</sub>, f<sub>HSI</sub>, f<sub>PCLK</sub>, f<sub>SYS</sub>.

### 25.5.1 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 198](#)).

In default configuration, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

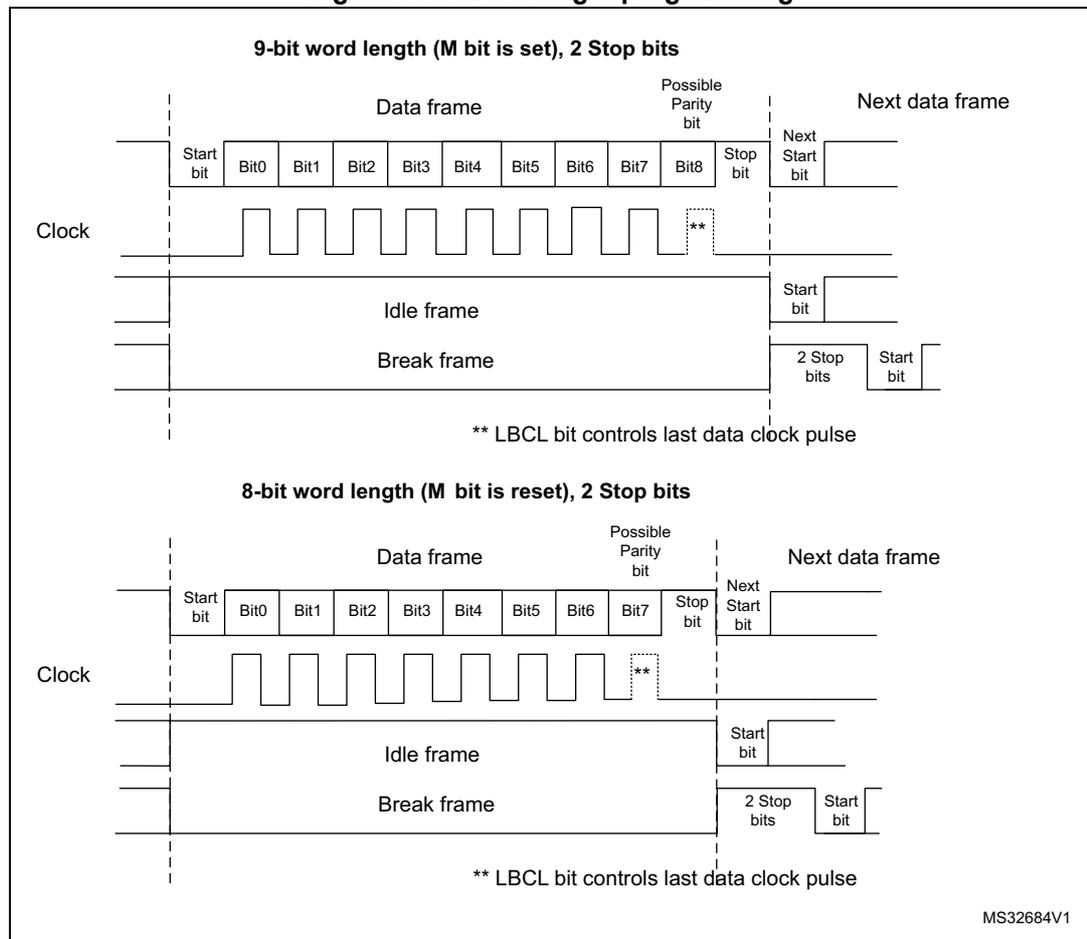
An **Idle character** is interpreted as an entire frame of “1”s followed by the start bit of the next frame which contains data (The number of “1” ‘s will include the number of stop bits).

A **Break character** is interpreted on receiving “0”s for a frame period. At the end of the break frame the transmitter inserts 2 stop bits.

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 198. Word length programming**



### 25.5.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the SCLK pin.

#### Character transmission

During an USART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the USART\_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see [Figure 197](#)).

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: 1, 1.5 and 2 stop bits.

*Note: The TE bit must be set before writing the data to be transmitted to the USART\_TDR. An idle frame will be sent after the TE bit is enabled.*

#### Configurable stop bits

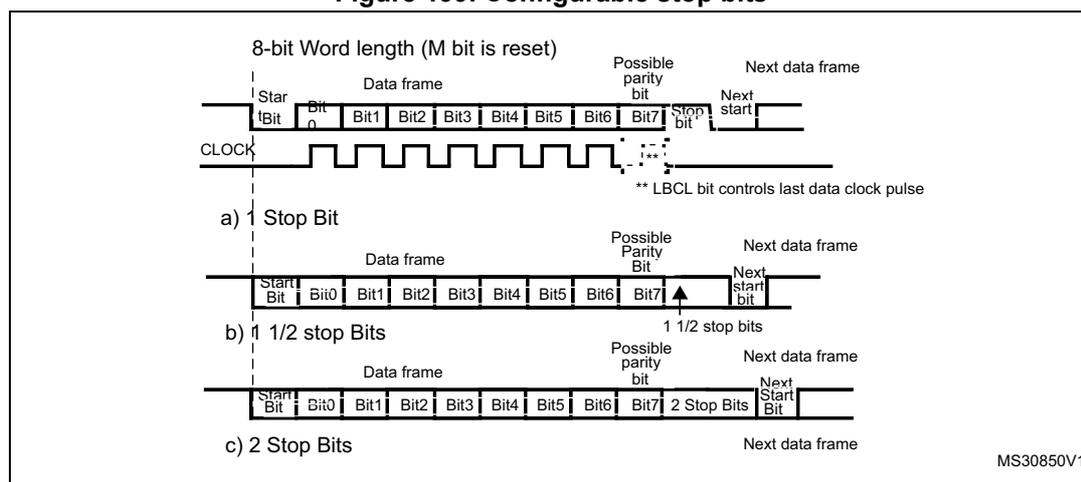
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

1. **1 stop bit:** This is the default value of number of stop bits.
2. **2 stop bits:** This will be supported by normal USART, single-wire and modem modes.
3. **1.5 stop bits:** To be used in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits (when m = 0) or 11 low bits (when m = 1) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 10/11 low bits).

Figure 199. Configurable stop bits



Procedure:

1. Program the M bit in USART\_CR1 to define the word length.
2. Select the desired baud rate using the USART\_BRR register.
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to 1.
5. Select DMA enable (DMAT) in USART\_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in USART\_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART\_TDR register (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\_TDR 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

Clearing the TXE bit is always performed by a write to the transmit data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from the USART\_TDR register to the shift register and the data transmission has started.
- The USART\_TDR register is empty.
- The next data can be written in the USART\_TDR 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\_TDR 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\_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

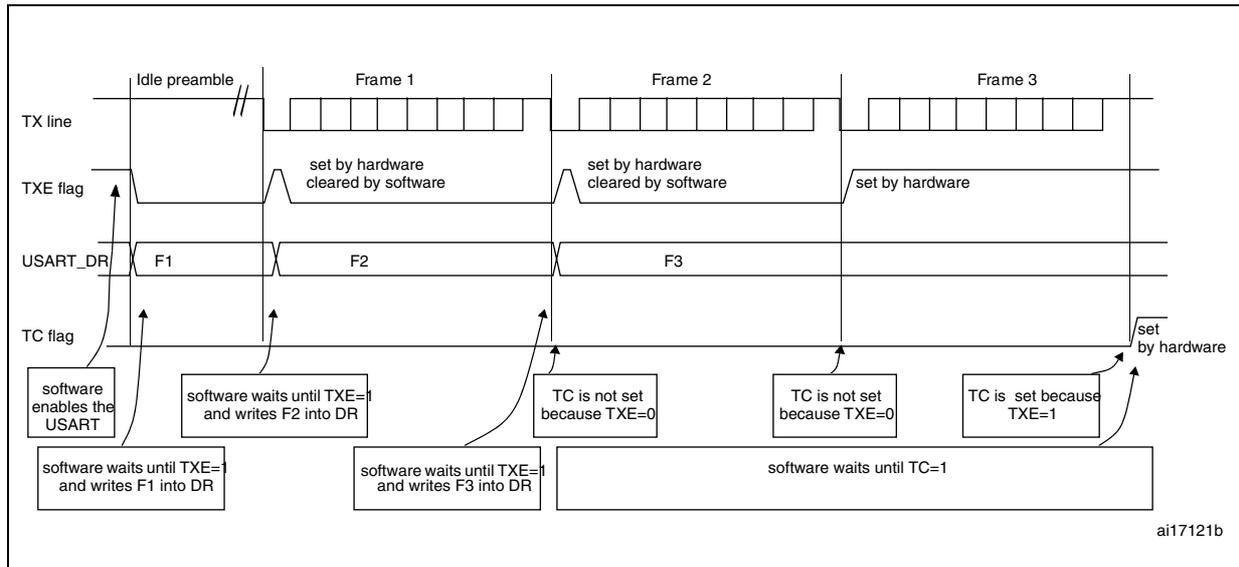
If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART\_CR1 register.

After writing the last data in the USART\_TDR 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 200: TC/TXE behavior when transmitting](#)).

**Note:** *The correct procedure to disable the USART is:*

- wait for TXE=1
- clear the TE bit (if a transmission is ongoing or a data word is in the USART\_TDR register, it will be sent before effectively disabling the transmission)
- the TC is set either immediately or at the end of the pending transmission
- clear the UE bit after TC=1

Figure 200. TC/TXE behavior when transmitting



**Break characters**

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see [Figure 198](#)).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

**Idle characters**

Setting the TE bit drives the USART to send an idle frame before the first data frame.

**25.5.3 Receiver**

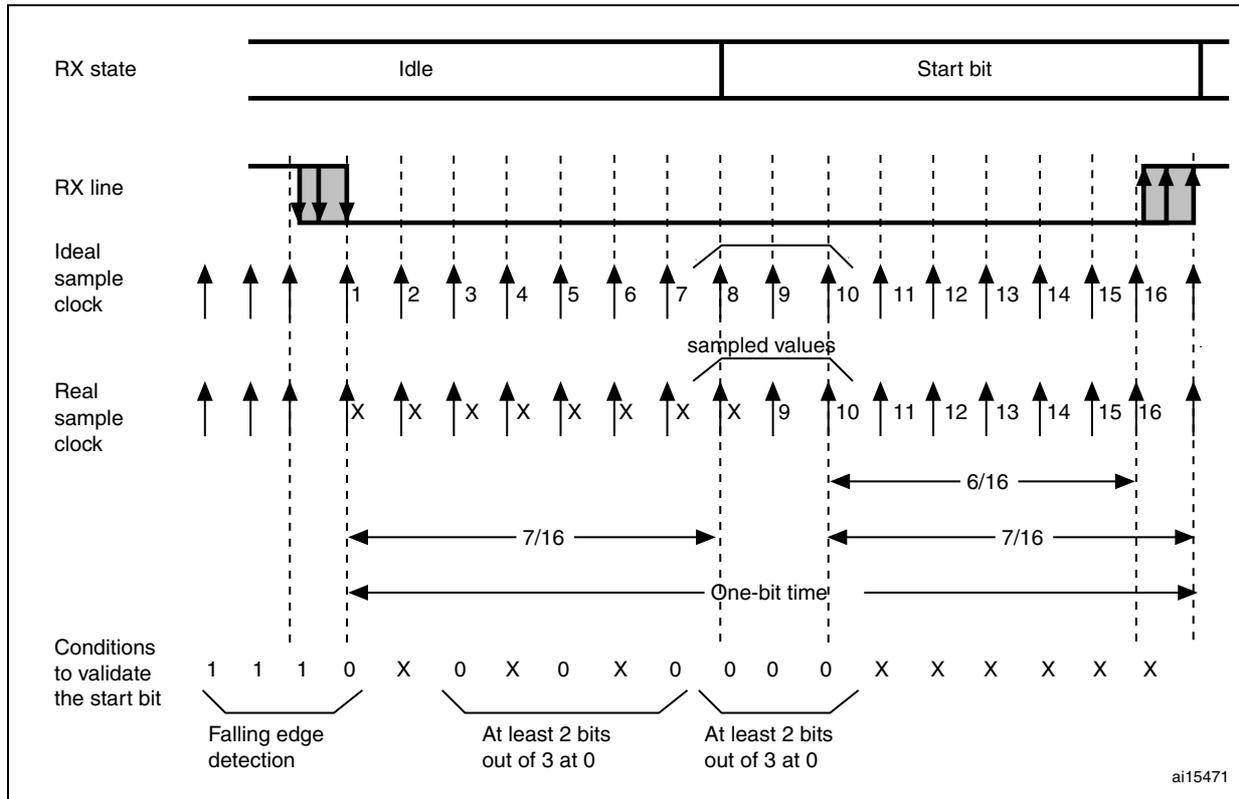
The USART can receive data words of either 8 or 9 bits depending on the M bit in the USART\_CR1 register.

**Start bit detection**

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 201. Start bit detection when oversampling by 16 or 8



**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,

a. for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b. for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither conditions a. or b. are met, the start detection aborts and the receiver returns to the idle state (no flag is set)

### Character reception

During an USART reception, data shifts in least significant bit first (default configuration) through the RX pin. In this mode, the USART\_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Procedure:

1. Program the M bit in USART\_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART\_BRR
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to 1.
5. Select DMA enable (DMAR) in USART\_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication. STEP 3
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. PE flag can also be set with RXNE.
- In multibuffer, RXNE is set after every byte received and is cleared by the DMA read of the Receive Data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART\_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

### 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 for a received data 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\_RDR 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 EIE bit is set.
- The ORE bit is reset by setting the ORECF bit in the ICR register.

**Note:** The ORE bit, when set, indicates that at least 1 data has been lost. 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.

### Selecting the clock source and the proper oversampling method

The choice of the clock source is done through the Clock Control system (see the [Reset and clock control \(RCC\) section](#)). The clock source must be chosen before enabling the USART (by setting the UE bit).

The choice of the clock source must be done according to two criteria:

- Possible use of the USART in low-power mode
- Communication speed

The clock source frequency is  $f_{CK}$ .

When the dual clock domain and the wakeup from Stop mode features are supported, the clock source can be one of the following sources:  $f_{PCLK}$  (default),  $f_{LSE}$ ,  $f_{HSI}$  or  $f_{SYS}$ . Otherwise, the USART clock source is  $f_{PCLK}$ .

Choosing  $f_{LSE}$ ,  $f_{HSI}$  as clock source may allow the USART to receive data while the MCU is in low-power mode. Depending on the received data and wakeup mode selection, the USART wakes up the MCU, when needed, in order to transfer the received data by software reading the USART\_RDR register or by DMA.

For the other clock sources, the system must be active in order to allow USART communication.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This allows a trade off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART\_CR1 register and can be either 16 or 8 times the baud rate clock ([Figure 202](#) and [Figure 203](#)).

Depending on the application:

- Select oversampling by 8 (OVER8=1) to achieve higher speed (up to  $f_{CK}/8$ ). In this case the maximum receiver tolerance to clock deviation is reduced (refer to [Section 25.5.5: Tolerance of the USART receiver to clock deviation on page 637](#))
- Select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum  $f_{CK}/16$

,where  $f_{CK}$  is the clock source frequency.

Programming the ONEBIT bit in the USART\_CR3 register selects the method used to evaluate the logic level. There are two options:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NF bit is set
- A single sample in the center of the received bit

Depending on the application:

- select the three samples' majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to [Figure 84](#)) because this indicates that a glitch occurred during the sampling.
- select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver's tolerance to clock deviations (see [Section 25.5.5: Tolerance of the USART receiver to clock deviation on page 637](#)). In this case the NF bit will never be set.

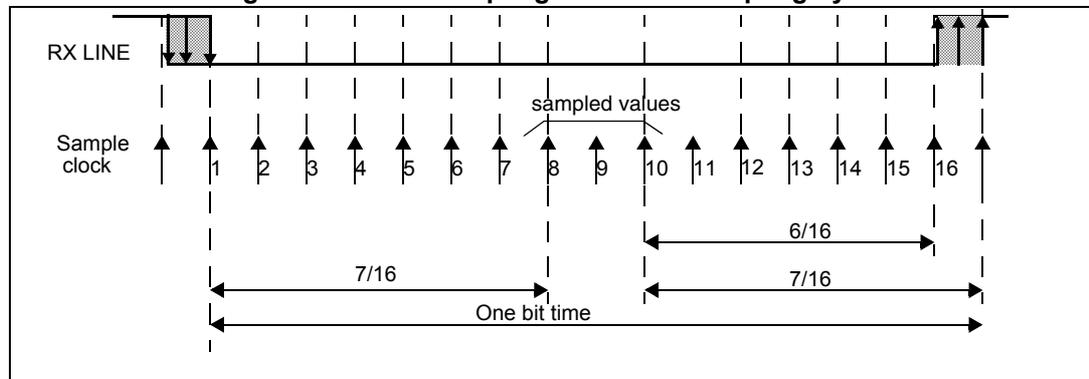
When noise is detected in a frame:

- The NF bit is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USART\_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit 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.

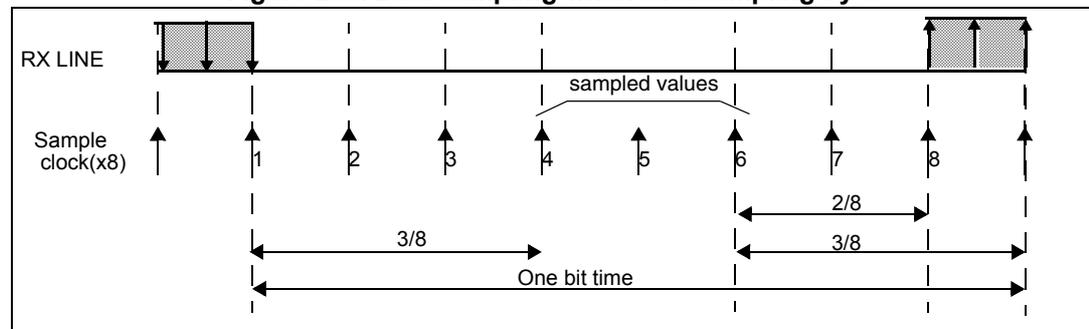
The NF bit is reset by setting NFCF bit in ICR register.

*Note: Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0 by hardware.*

**Figure 202. Data sampling when oversampling by 16**



**Figure 203. Data sampling when oversampling by 8**



**Table 84. Noise detection from sampled data**

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

### Framing error

A framing error is detected when:

The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the USART\_RDR register.
- 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 writing 1 to the FE CF in the USART\_ICR register.

### 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 1.5 in Smartcard mode.

1. **1 stop bit:** Sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
2. **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 25.5.13: Smartcard mode on page 648](#) for more details.
3. **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.

## 25.5.4 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 USART\_BRR register.

### Equation 1: Baud rate for standard USART (SPI mode included)

In case of oversampling by 16, the equation is:

$$\text{Tx/Rx baud} = \frac{f_{\text{CK}}}{\text{USARTDIV}}$$

In case of oversampling by 8, the equation is:

$$\text{Tx/Rx baud} = \frac{2 \times f_{\text{CK}}}{\text{USARTDIV}}$$

### Equation 2: Baud rate in Smartcard, LIN and IrDA modes

In Smartcard, LIN and IrDA modes, only Oversampling by 16 is supported:

$$\text{Tx/Rx baud} = \frac{f_{\text{CK}}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART\_BRR register.

- When OVER8=0, BRR = USARTDIV
- When OVER8=1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR3 must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

*Note:* The baud counters are updated to the new value in the baud registers after a write operation to USART\_BRR. Hence the baud rate register value should not be changed during communication.

*In case of oversampling by 16 or 8, USARTDIV must be greater than or equal to 16d.*

### How to derive USARTDIV from USART\_BRR register values when OVER8=0

#### Example 1

To obtain 9600 baud with  $f_{\text{CK}} = 8 \text{ MHz}$ .

- In case of oversampling by 16:
  - USARTDIV =  $8\,000\,000/9600$
  - BRR[31:0] = USARTDIV = 833d = 0x0341
- In case of oversampling by 8:
  - USARTDIV =  $2 * 8\,000\,000/9600$
  - USARTDIV = 1666,66 (1667d = 0x683)
  - BRR[3:0] =  $0x3 \gg 1 = 0x1$
  - BRR = 0x681

#### Example 2

To obtain 921.6 Kbaud with  $f_{CK} = 48 \text{ MHz}$ .

- In case of oversampling by 16:  
 $USARTDIV = 48\ 000\ 000/921\ 600$   
 $BRR = USARTDIV = 52d = 0x34$
- In case of oversampling by 8:  
 $USARTDIV = 2 * 48\ 000\ 000/921\ 600$   
 $USARTDIV = 104 (104d = 0x68)$   
 $BRR[3:0] = USARTDIV[3:0] \gg 1 = 0x8 \gg 1 = 0x4$   
 $BRR = 0x64$

**Table 85. Error calculation for programmed baud rates at  $f_{CK} = 72 \text{ MHz}$  for oversampling by 16 and by 8**

Baudrate		Oversampling by 16 (OVER8 = 0)			Oversampling by 8 (OVER8 = 1)		
S.No	Desired	Actual	USART_BRR value	% Error = (calculated - Desired baudrate)/Desired baudrate	Actual	USART_BRR value	% Error = (calculated - Desired baudrate)/Desired baudrate
2	2.4	2.4	0x7530	0	2.4	0xEA60	0
3	9.6	9.6	0x1D4C	0	9.6	0x3A94	0
4	19.2	19.2	0xEA6	0	19.2	0x1D46	0
5	38.4	38.4	0x753	0	38.4	0xEA3	0
6	57.6	57.6	0x4E2	0	57.6	0x9C2	0
7	115.2	115.2	0x271	0	115.2	0x4E1	0
8	230.4	230.03	0x139	0.16	230.4	0x270	0
9	460.8	461.54	0x9C	0.16	460.06	0x134	0.16
10	921.6	923.08	0x4E	0.16	923.07	0x96	0.16
11	2000	2000	0x24	0	2000	0x44	0
12	3000	3000	0x18	0	3000	0x30	0
13	4000	4000	0x12	0	4000	0x22	0
14	5000	N.A	N.A	N.A	4965.51	0x16	0.69
15	6000	N.A	N.A	N.A	6000	0x14	0
16	7000	N.A	N.A	N.A	6857.14	0x12	2
17	9000	N.A	N.A	N.A	9000	0x10	0

### 25.5.5 Tolerance of the USART receiver to clock deviation

The asynchronous receiver of the USART works correctly only if the total clock system deviation is less than the tolerance of the USART receiver. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver’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)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver’s tolerance}$$

Where

DWU is the error due to sampling point deviation when the wakeup from Stop mode is used.

when M = 1:

$$DWU = \frac{t_{WUSTOP}}{(11 \times Tbit)}$$

when M=0:

$$DWU = \frac{t_{WUSTOP}}{(10 \times Tbit)}$$

$t_{WUSTOP}$  is the wakeup time from Stop mode, which is specified in the product datasheet.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in [Table 86](#) and [Table 87](#), depending on the following choices:

- 10- or 11-bit character length defined by the M bit in the USART\_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART\_CR1 register
- Bits BRR[3:0] of USART\_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART\_CR3 register

**Table 86. Tolerance of the USART receiver when BRR[3:0] = 0000**

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.75%	4.375%	2.50%	3.75%
1	3.41%	3.97%	2.27%	3.41%

**Table 87. Tolerance of the USART receiver when BRR[3:0] is different from 0000**

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.33%	3.88%	2%	3%
1	3.03%	3.53%	1.82%	2.73%

*Note:* The data specified in [Table 86](#) and [Table 87](#) may slightly 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).

### 25.5.6 Auto baud rate detection

The USART is able to detect and automatically set the USART\_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance
- The system is using a relatively low accuracy clock source and this mechanism allows the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed (When oversampling by 16, the baud rate is between  $f_{CK}/65535$  and  $f_{CK}/16$ . When oversampling by 8, the baudrate is between  $f_{CK}/65535$  and  $f_{CK}/8$ ).

Before activating the auto baud rate detection, the auto baud rate detection mode must be chosen. There are four modes ( based on different character patterns) which can be chosen through the ABRMOD[1:0] field in the USART\_CR2 register. In these 4 auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are:

1. Mode 0: Any character starting with a bit at 1. In this case the USART measures the duration of the start bit (falling edge to rising edge).
2. Mode 1: Any character starting with a 10xx bit pattern. In this case, the USART measures the duration of the start and of the 1st data bit. The measurement is done falling edge to falling edge, ensuring better accuracy in the case of slow signal slopes.
3. Mode 2: 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode). In this case, the baudrate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit 0 to Bit 6 are sampled at BRs, further bits of the character are sampled at BR6.
4. Mode 3: A 0x55 character frame. In this case, the baudrate is updated first at the end of the start bit (BRs), then at the end of bit 0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit 6 (BR6). Bit 0 is sampled at BRs, Bit 1 to Bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate transition of RX line. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating auto baud rate detection, the USART\_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART\_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART\_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The RXNE interrupt will signal the end of the operation.

At any later time, the auto baud rate detection may be relaunched by resetting the ABRF flag (by writing a 0).

*Note: If the USART is disabled (UE=0) during an auto baud rate operation, the BRR value may be corrupted.*

## 25.5.7 Multiprocessor communication

It is possible to perform multiprocessor communication with the USART (with several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output connected to the RX inputs of the other USARTs. 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 order to use the mute mode feature, the MME bit must be set in the USART\_CR1 register.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART\_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART\_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART\_CR1 register:

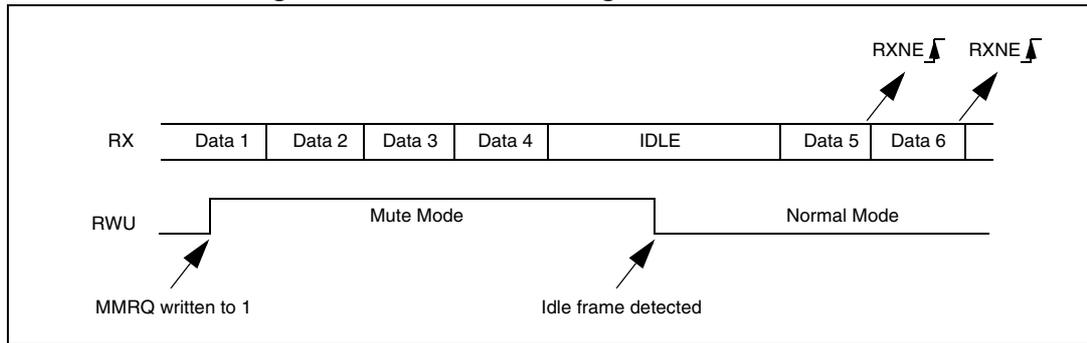
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

### Idle line detection (WAKE=0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

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\_ISR register. An example of mute mode behavior using Idle line detection is given in [Figure 204](#).

Figure 204. Mute mode using Idle line detection



*Note:* If the MMRQ is set while the IDLE character has already elapsed, mute mode will not be entered (RWU is not set).  
 If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

**4-bit/7-bit address mark detection (WAKE=1)**

In this mode, bytes are recognized as addresses if their MSB is a '1' otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART\_CR2 register.

*Note:* In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

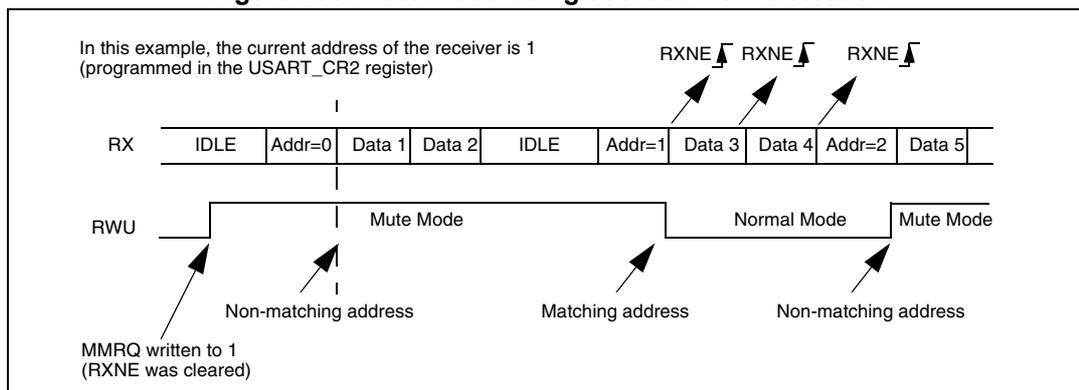
The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode.

The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit is set for the address character since the RWU bit has been cleared.

An example of mute mode behavior using address mark detection is given in [Figure 205](#).

Figure 205. Mute mode using address mark detection



### 25.5.8 ModBus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half duplex, block transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

#### Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART\_CR2 register and the RTOIE in the USART\_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

#### Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE=1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

### 25.5.9 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 88](#).

Table 88. Frame formats

M bit	PCE bit	USART frame <sup>(1)</sup>
0	0	SB   8-bit data   STB
0	1	SB   7-bit data   PB   STB
1	0	SB   9-bit data   STB
1	1	SB   8-bit data   PB   STB

- Legends: SB: start bit, STB: stop bit, PB: parity bit.
- In the data register, the PB is always taking the MSB position (8<sup>th</sup> or 9<sup>th</sup>, depending on the M bit value).

**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.

E.g.: 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.

E.g.: data=00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PS bit in USART\_CR1 = 1).

**Parity checking in reception**

If the parity check fails, the PE flag is set in the USART\_ISR register and an interrupt is generated if PEIE is set in the USART\_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART\_ICR register.

**Parity generation in transmission**

If the PCE bit is set in USART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

**25.5.10 LIN (local interconnection network) mode**

This section is relevant only when LIN mode is supported. Please refer to [Section 25.4: USART implementation on page 623](#).

The LIN mode is selected by setting the LINEN bit in the USART\_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0], CLKEN in the USART\_CR2 register
- SCEN, HDSEL and IREN in the USART\_CR3 register.

### LIN transmission

The procedure explained in [Section 25.5.2](#) has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 '0 bits as a break character. Then 2 bits of value '1 are sent to allow the next start detection.

### LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART\_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART\_CR2) or 11 (when LBDL=1 in USART\_CR2) consecutive bits are detected as '0, and are followed by a delimiter character, the LBDF flag is set in USART\_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a '1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

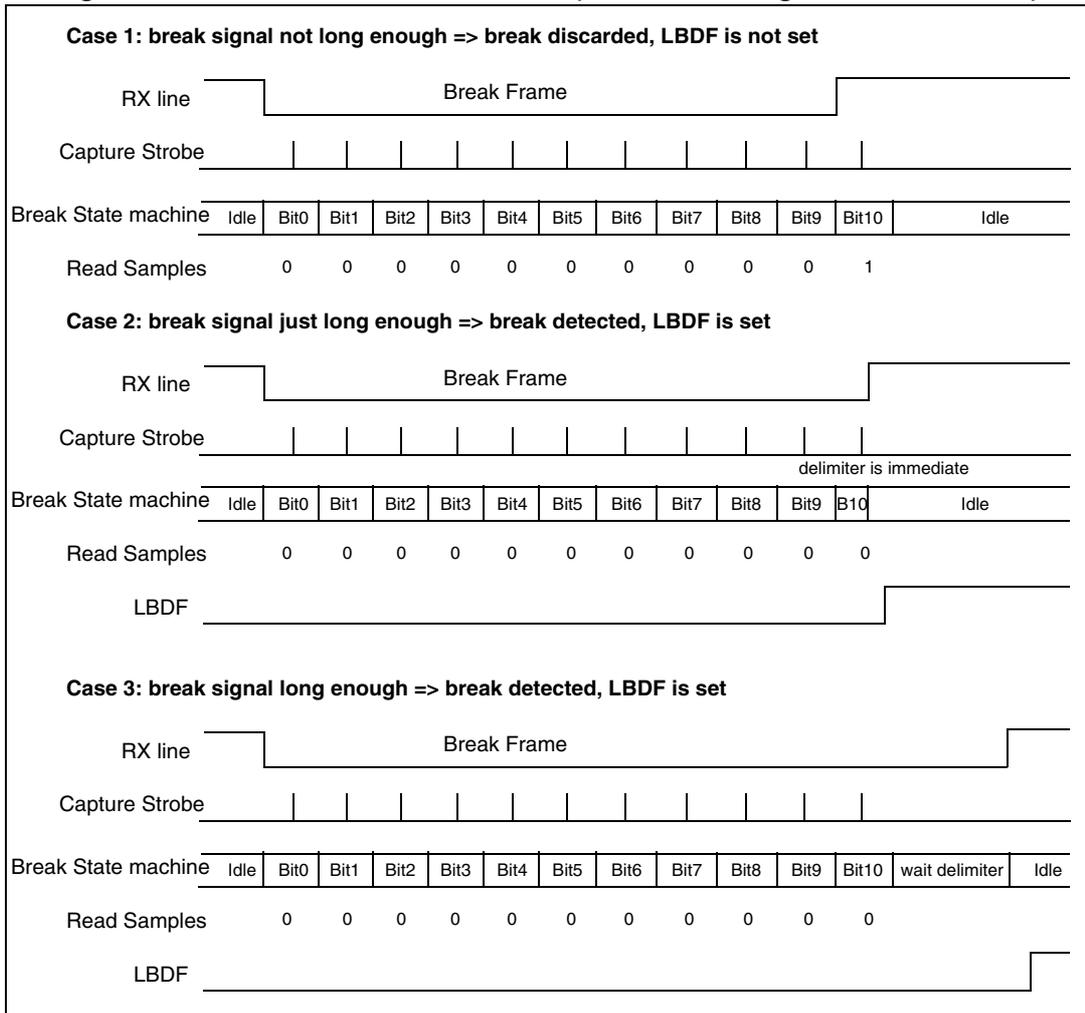
If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at '0, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a '1, if the break word was not complete, or a delimiter character if a break has been detected.

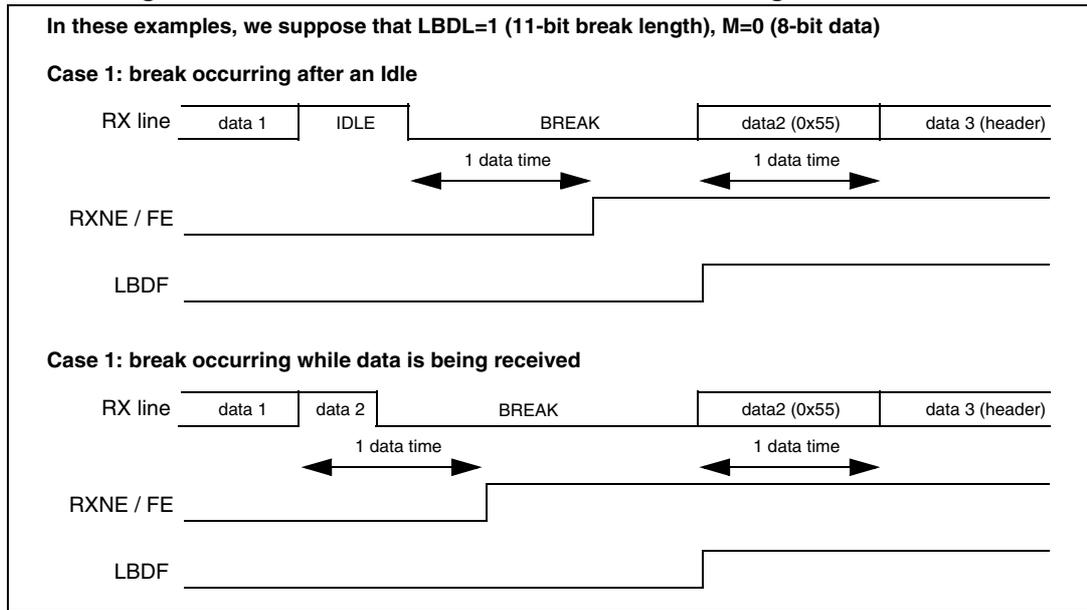
The behavior of the break detector state machine and the break flag is shown on the [Figure 206: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 644](#).

Examples of break frames are given on [Figure 207: Break detection in LIN mode vs. Framing error detection on page 645](#).

**Figure 206. Break detection in LIN mode (11-bit break length - LBDF bit is set)**



**Figure 207. Break detection in LIN mode vs. Framing error detection**



### 25.5.11 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.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART\_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART\_CR2 register is used to select the clock polarity, and the CPHA bit in the USART\_CR2 register is used to select the phase of the external clock (see [Figure 208](#), [Figure 209](#) & [Figure 210](#)).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as SCLK 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 SCLK (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:* The SCLK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data is being transmitted (the data register USART\_TDR written). This means that it is not possible to receive synchronous data without transmitting data.

The LBCL, CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

Figure 208. USART example of synchronous transmission

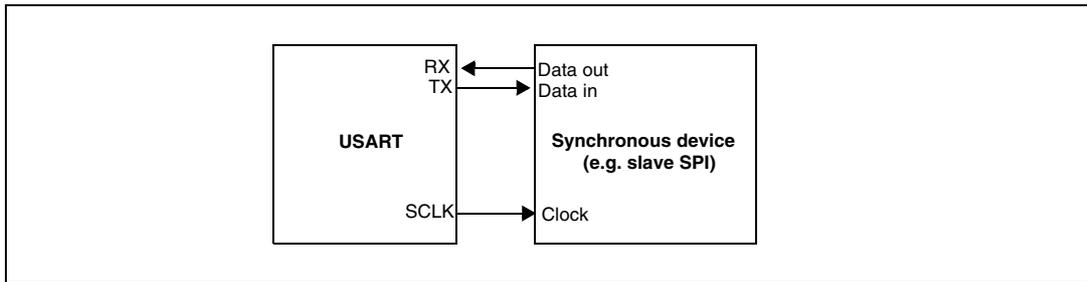


Figure 209. USART data clock timing diagram (M=0)

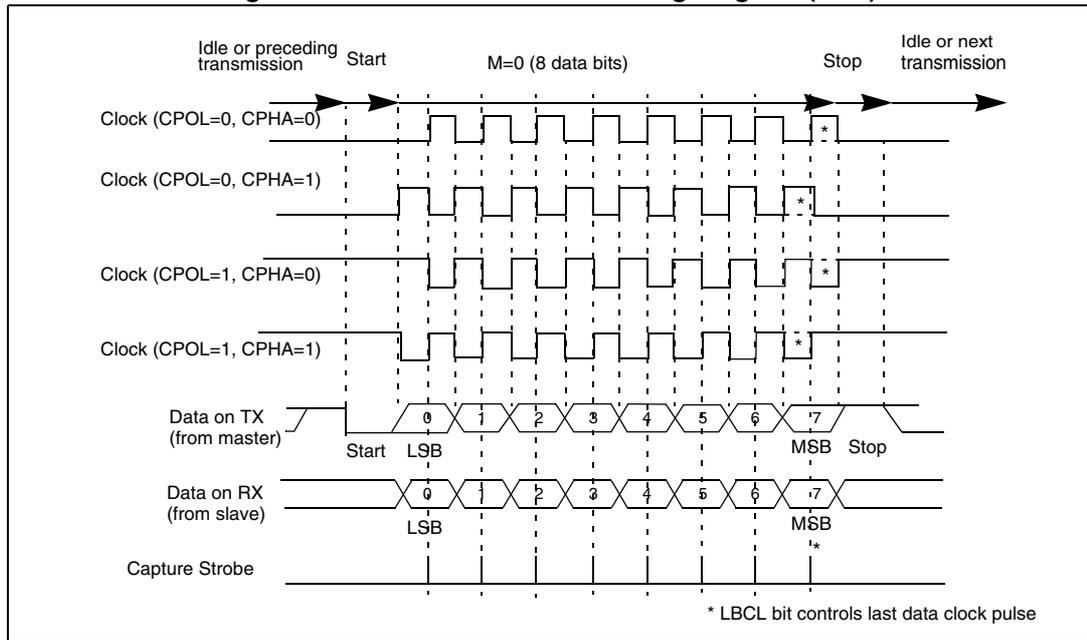


Figure 210. USART data clock timing diagram (M=1)

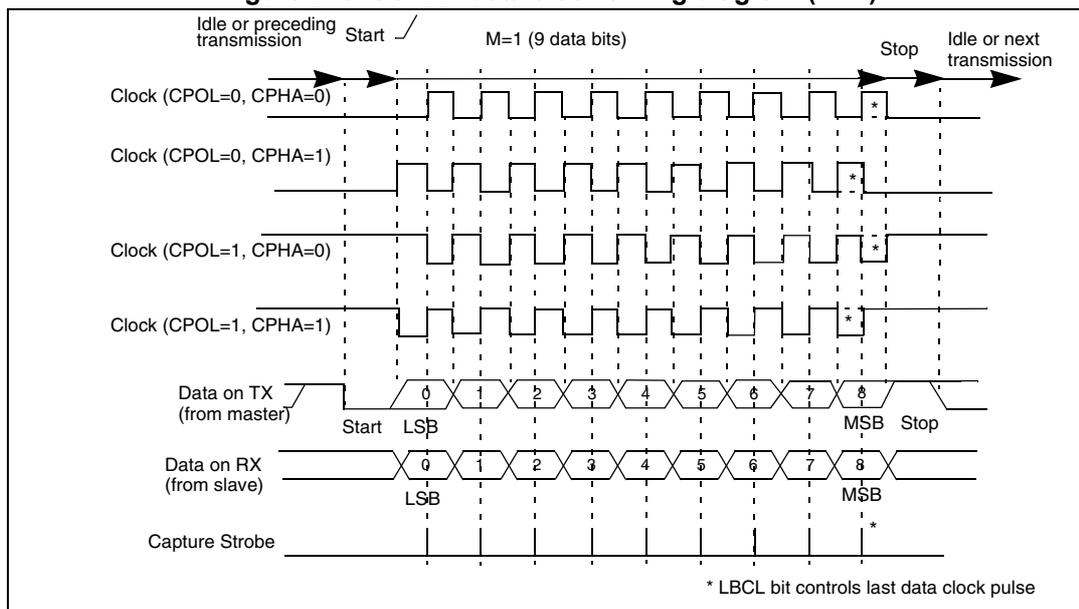
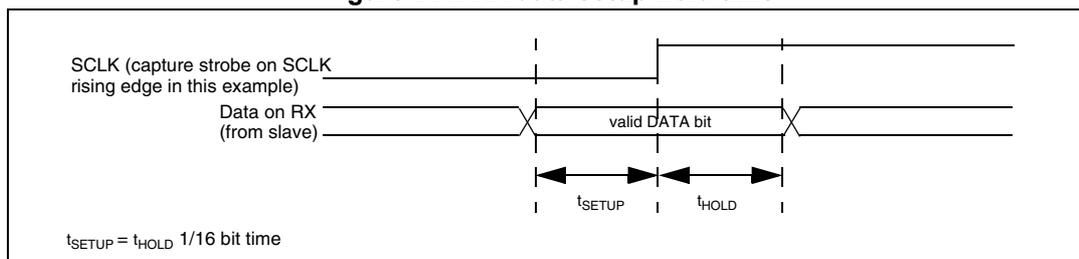


Figure 211. RX data setup/hold time



Note: The function of SCLK is different in Smartcard mode. Refer to [Section 25.5.13: Smartcard mode](#) for more details.

### 25.5.12 Single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART\_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART\_CR2 register,
- SCEN and IREN bits in the USART\_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit 'HALF DUPLEX SEL' (HDSEL in USART\_CR3).

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflicts on the line must be managed by software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

### 25.5.13 Smartcard mode

This section is relevant only when Smartcard mode is supported. Please refer to [Section 25.4: USART implementation on page 623](#).

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. Both T=0 (character mode) and T=1 (block mode) are supported.

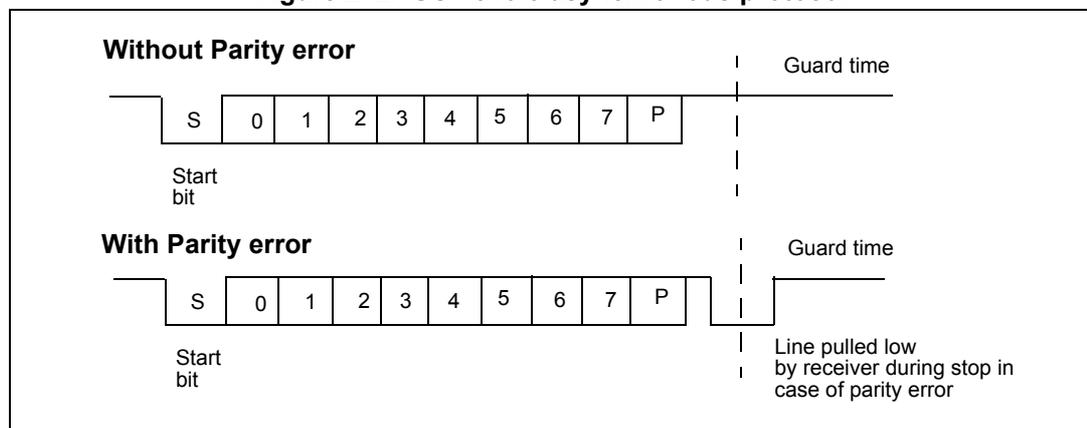
The USART should be configured as:

- 8 bits plus parity: where M=1 and PCE=1 in the USART\_CR1 register
- 1.5 stop bits: where STOP=11 in the USART\_CR2 register.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

[Figure 212](#) shows examples of what can be seen on the data line with and without parity error.

**Figure 212. ISO 7816-3 asynchronous protocol**



When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts

shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.

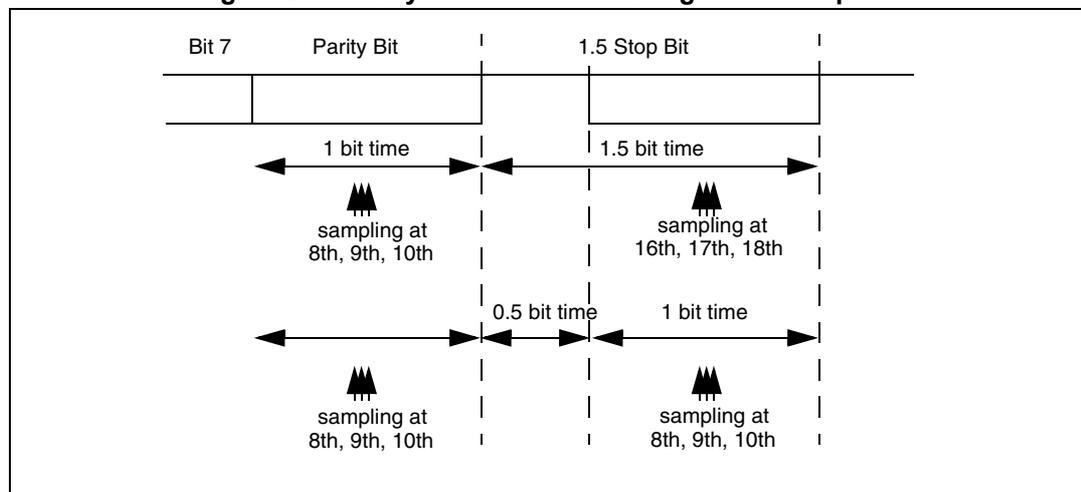
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol. The number of retries is programmed in the SCARCNT bit field. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit may be cleared using the TXFRQ bit in the USART\_RQR register.
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T=1 mode). If the received character is erroneous, the RXNE/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bit field, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

*Note:* A break character is not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

*No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.*

Figure 213 details how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 213. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the SCLK output. In Smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART\_GTPR. SCLK frequency can be programmed from  $f_{CK}/2$  to  $f_{CK}/62$ , where  $f_{CK}$  is the peripheral input clock.

Block mode (T=1)

In T=1 (block) mode, the parity error transmission is deactivated, by clearing the NACK bit in the UART\_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt will be generated. If the first character is received before the expiration of the period, it is signaled by the RXNE interrupt.

*Note: The RXNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.*

After the reception of the first character (RXNE interrupt), the RTO register must be programmed to the CWT (character wait time) - 11 value, in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baudtime units. If the Smartcard doesn't send a new character in less than the CWT period after the end of the previous character, the USART signals this to the software through the RTOF flag and interrupt (when RTOIE bit is set).

*Note: The RTO counter starts counting from the end of the first stop bit of the last character in cases STOP = 00, 10. In case of STOP = 11, the RTO counter starts counting 1 bit time after the beginning of the STOP bit. As in the Smartcard protocol definition, the BWT/CWT values are defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT -11 or CWT -11, respectively, taking into account the length of the last character itself.*

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting (TXE=0). The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART\_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value (0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBFF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

*Note:* The error checking code (LRC/CRC) must be computed/verified by software.

### Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

*Note:* When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character will be '03' and the parity will be odd.

Therefore, two methods are available for TS pattern recognition:

**Method 1:** The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card didn't answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it will be correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

**Method 2:** The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHH LLL LLH = 0x103 -> inverse convention to be chosen

(H) LHHH HHH LLH = 0x13B -> direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

#### 25.5.14 IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Please refer to [Section 25.4: USART implementation on page 623](#).

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 214](#)).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not

encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see [Figure 215](#)).
- 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  $\mu\text{s}$ . The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART\_GTPR register). 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 ( $1.42 \text{ MHz} < \text{PSC} < 2.12 \text{ MHz}$ ). A low-power mode programmable divisor divides the system clock to achieve this value.

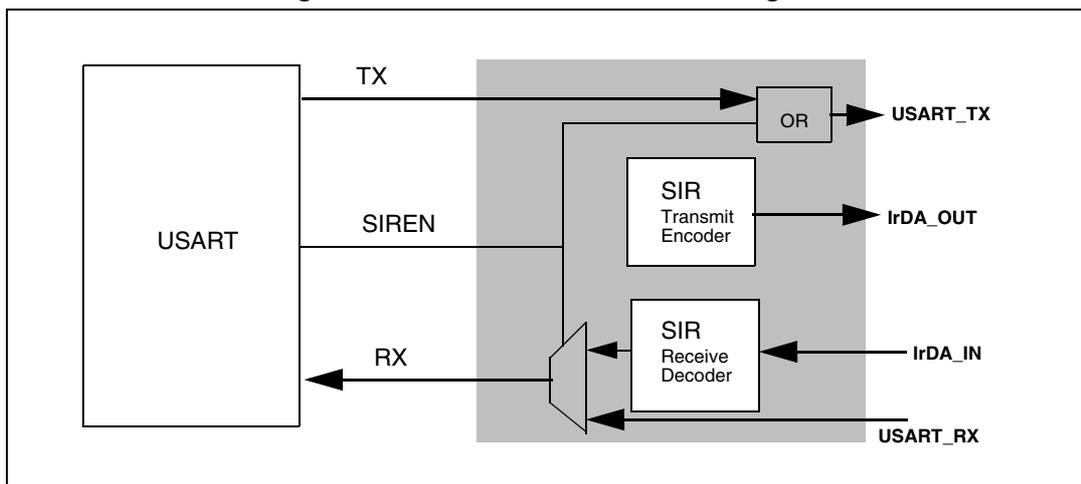
**Receiver:**

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than  $1/\text{PSC}$ . A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART\_GTPR register).

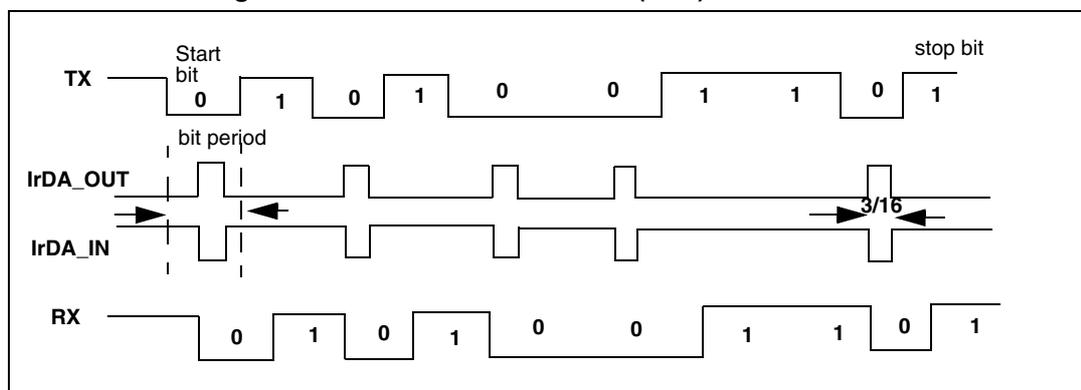
*Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.*

*The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).*

**Figure 214. IrDA SIR ENDEC- block diagram**



**Figure 215. IrDA data modulation (3/16) -Normal Mode**



## 25.5.15 Continuous communication using DMA

The USART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

*Note:* Please refer to [Section 25.4: USART implementation on page 623](#) to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in [Section 25.5.2](#) or [25.5.3](#). To perform continuous communication, you can clear the TXE/ RXNE flags in the USART\_ISR register.

### Transmission using DMA

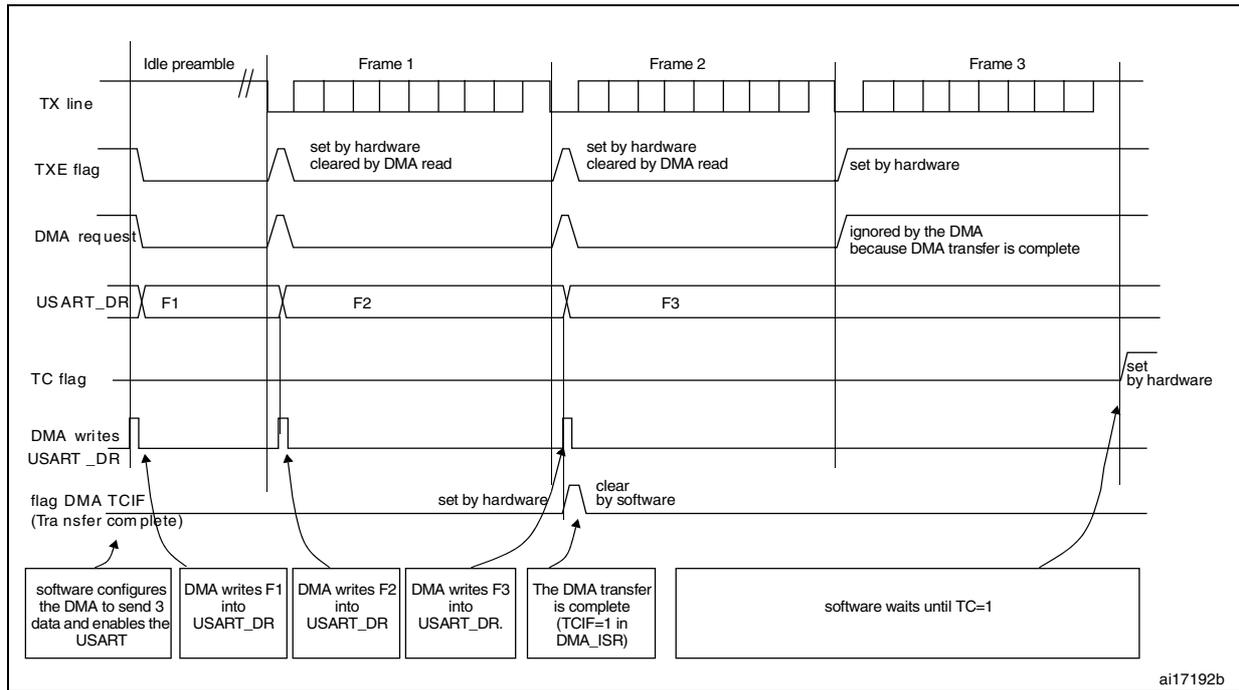
DMA mode can be enabled for transmission by setting DMAT bit in the USART\_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to [Section 10: Direct memory access controller \(DMA\) on page 162](#)) to the USART\_TDR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART\_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART\_TDR register from this memory area after each TXE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART\_ISR register by setting the TCCF bit in the USART\_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering Stop mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 216. Transmission using DMA



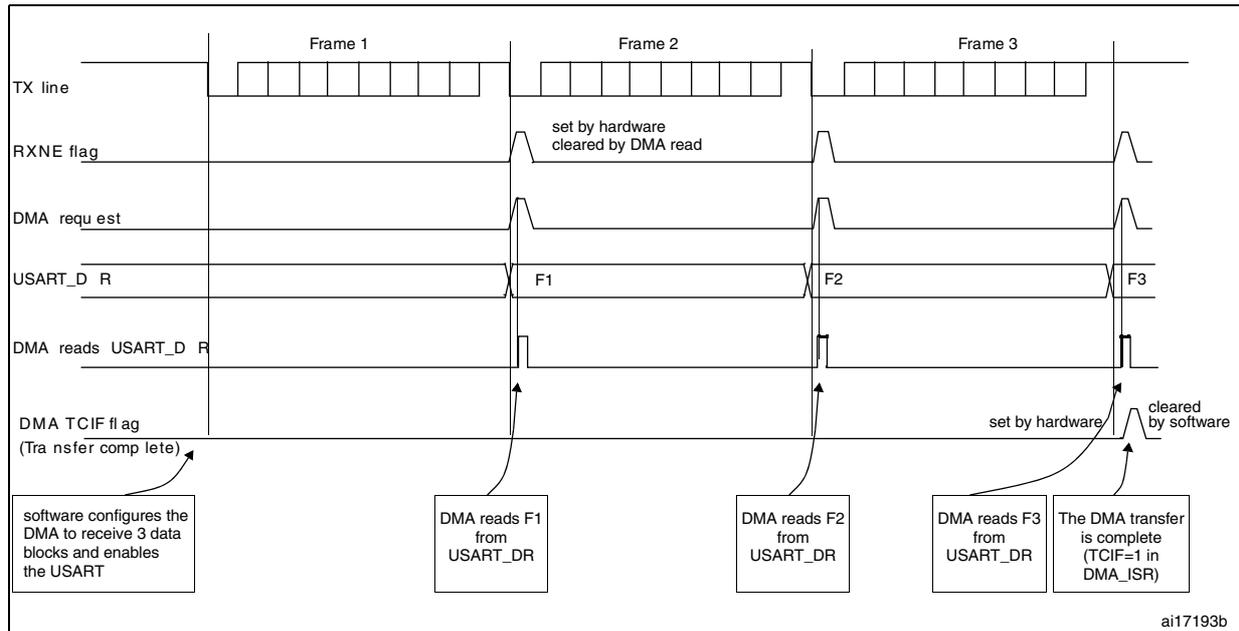
### Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART\_CR3 register. Data is loaded from the USART\_RDR register to a SRAM area configured using the DMA peripheral (refer [Section 10: Direct memory access controller \(DMA\) on page 162](#)) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART\_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART\_RDR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 217. Reception using DMA



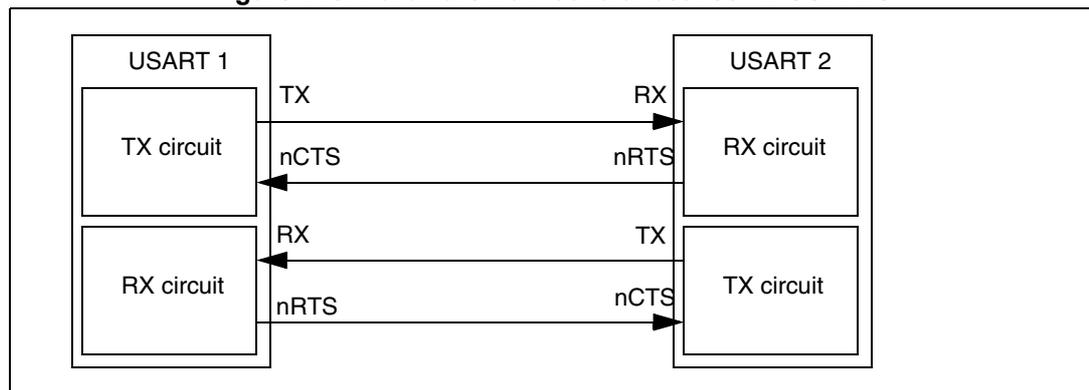
### Error flagging and interrupt generation in multibuffer communication

In multibuffer communication if any error occurs during the transaction the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART\_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

### 25.5.16 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The Figure 218 shows how to connect 2 devices in this mode:

Figure 218. Hardware flow control between 2 USARTs

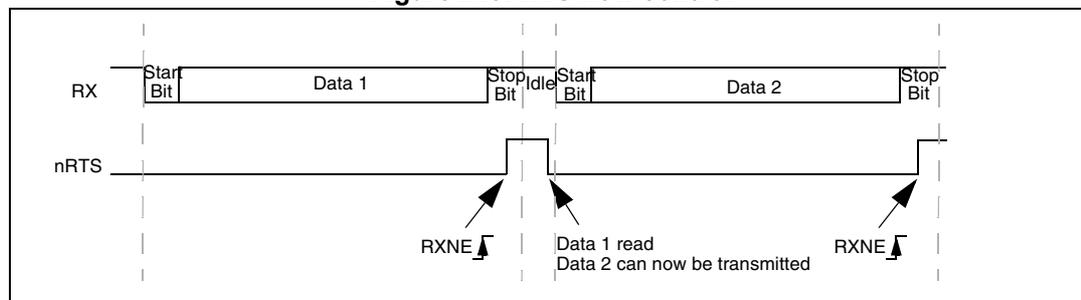


RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the USART\_CR3 register).

### 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 a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 219* shows an example of communication with RTS flow control enabled.

Figure 219. RTS flow control

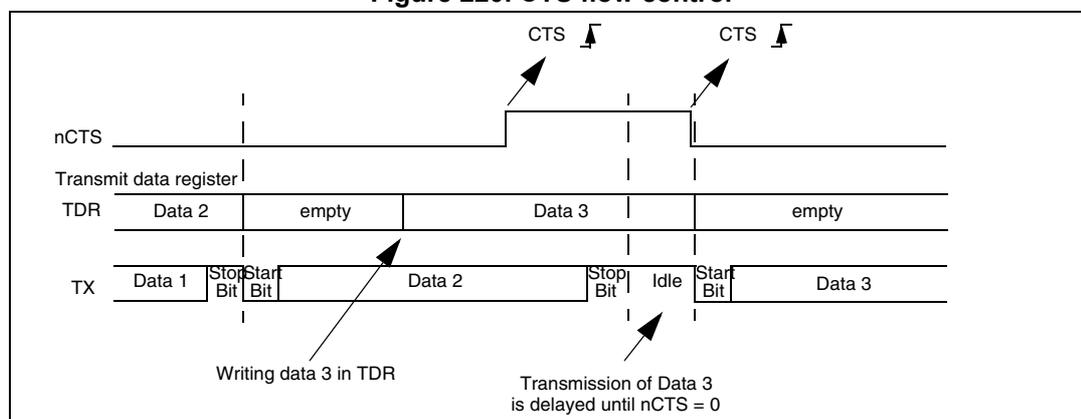


### CTS flow control

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 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. *Figure 220* shows an example of communication with CTS flow control enabled.

Figure 220. CTS flow control



**Note:** For correct behavior, nCTS must be asserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

### RS485 Driver Enable

The driver enable feature is enabled by setting bit DEM in the USART\_CR3 control register. This allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the START bit. It is programmed using the DEAT [4:0] bit fields in the USART\_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bit fields in the USART\_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART\_CR3 control register.

### 25.5.17 Wakeup from Stop mode

This section is relevant only when the Wakeup from Stop feature is supported. Please refer to [Table 83: USART features on page 623](#).

The USART is able to wake up the MCU from Stop mode when the UESM bit is set and the USART clock is set to HSI or LSE (refer to the [Reset and clock control \(RCC\) section](#)).

The MCU wakeup from Stop mode can be done using the standard RXNE interrupt. In this case, the RXNEIE bit must be set before entering Stop mode.

Alternatively, a specific interrupt may be selected through the WUS bit fields.

In order to be able to wake up the MCU from Stop mode, the UESM bit in the USART\_CR1 control register must be set prior to entering Stop mode.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUFIE bit is set.

**Note:** *Before entering Stop mode, software must check that the USART is not performing a transfer, by checking the BUSY flag in the USART\_SR register.*

*The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in Stop or in an active mode.*

*When entering Stop mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the USART is actually enabled.*

*The minimum bit duration to wake up properly from Stop mode is  $t_{WUSTOP} / (DWU (max) * 11)$  when M equals 1 and  $t_{WUSTOP} / (DWU (max) * 10)$  when M equals 0.*

**Caution:** The wakeup from Stop mode feature is not available for all modes. For example it doesn't work in SPI mode because the SPI operates in master mode only.

### Using Mute mode with Stop mode

If the USART is put into Mute mode before entering Stop mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in Stop mode.
- If the wakeup from Mute mode on address match is used, then the source of wake-up from Stop mode must also be the address match.
- If the USART is configured to wake up the MCU from Stop mode on START bit detection, the WUF flag is set, but the RXNE flag is not set.

## 25.6 USART interrupts

Table 89. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TXE	TXEIE
CTS interrupt	CTSIF	CTSIE
Transmission Complete	TC	TCIE
Receive data register not empty (data ready to be read)	RXNE	RXNEIE
Overrun error detected	ORE	
Idle line detected	IDLE	IDLEIE
Parity error	PE	PEIE
LIN break	LBDF	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication.	NF or ORE or FE	EIE
Character match	CMF	CMIE
Receiver timeout error	RTOF	RTOIE
End of Block	EOBF	EOBIE
Wakeup from Stop mode	WUF <sup>(1)</sup>	WUFIE

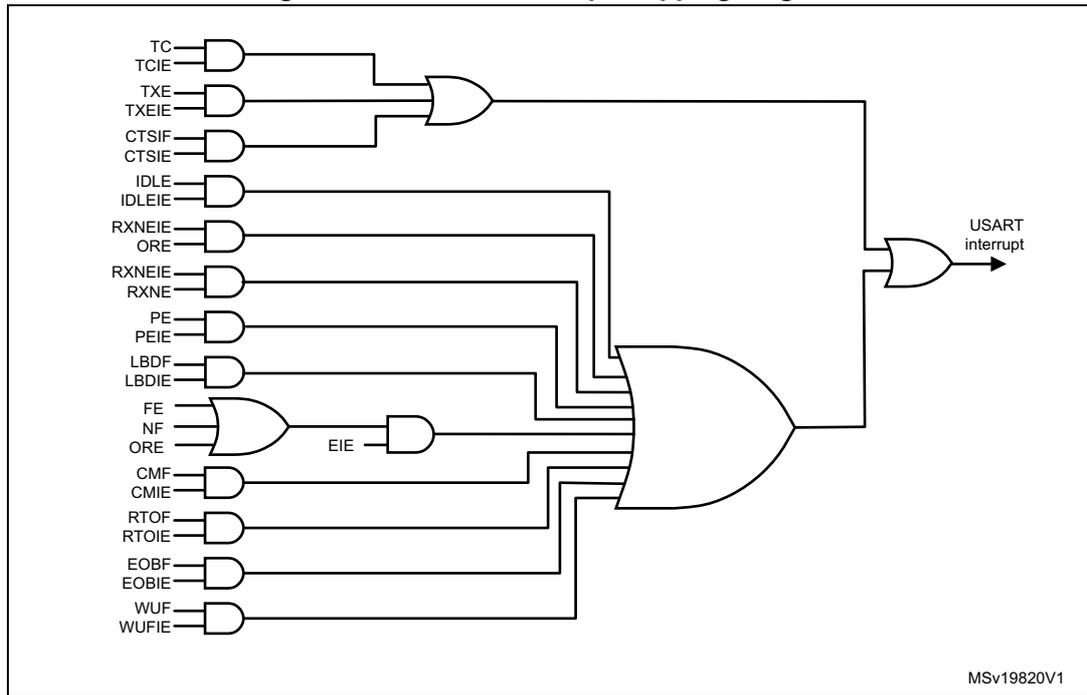
1. The WUF interrupt is active only in Stop mode.

The USART interrupt events are connected to the same interrupt vector (see [Figure 221](#)).

- During transmission: Transmission Complete, Clear to Send, Transmit Data Register empty or Framing error (in Smartcard mode) interrupt.
- During reception: Idle Line detection, Overrun error, Receive data register not empty, Parity error, LIN break detection, Noise Flag (only in multi buffer communication), Framing Error (only in multi buffer communication), Character match, etc.

These events generate an interrupt if the corresponding Enable Control Bit is set.

Figure 221. USART interrupt mapping diagram



## 25.7 USART registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

### 25.7.1 Control register 1 (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER 8	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEI E	IDLEIE	TE	RE	UESM	UE
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the EOBIF flag is set in the USART\_ISR register

*Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A USART interrupt is generated when the RTOF bit is set in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. [Section 25.4: USART implementation on page 623](#).*

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate)

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Please refer to [Section 25.4: USART implementation on page 623](#).*

This bit field can only be written when the USART is disabled (UE=0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Please refer to [Section 25.4: USART implementation on page 623](#).*

This bit field can only be written when the USART is disabled (UE=0).

- Bit 15 **OVER8**: Oversampling mode  
0: Oversampling by 16  
1: Oversampling by 8  
*Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.*  
This bit can only be written when the USART is disabled (UE=0).
- Bit 14 **CMIE**: Character match interrupt enable  
This bit is set and cleared by software.  
0: Interrupt is inhibited  
1: A USART interrupt is generated when the CMF bit is set in the USART\_ISR register.
- Bit 13 **MME**: Mute mode enable  
This bit activates the mute mode function of the USART. When set, the USART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.  
0: Receiver in active mode permanently  
1: Receiver can switch between mute mode and active mode.
- Bit 12 **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  
This bit can only be written when the USART is disabled (UE=0).
- Bit 11 **WAKE**: Receiver wakeup method  
This bit determines the USART wakeup method from Mute mode. It is set or cleared by software.  
0: Idle line  
1: Address mark  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 10 **PCE**: Parity control enable  
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).  
0: Parity control disabled  
1: Parity control enabled  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 9 **PS**: Parity selection  
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity will be selected after the current byte.  
0: Even parity  
1: Odd parity  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 8 **PEIE**: PE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt is inhibited  
1: A USART interrupt is generated whenever PE=1 in the USART\_ISR register

- Bit 7 **TXEIE**: 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\_ISR 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\_ISR 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\_ISR 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\_ISR register
- Bit 3 **TE**: Transmitter enable  
This bit enables the transmitter. It is set and cleared by software.  
0: Transmitter is disabled  
1: Transmitter is enabled

*Note:* 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. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the USART\_ISR register.  
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 UESM:** USART enable in Stop mode

When this bit is cleared, the USART is not able to wake up the MCU from Stop mode.

When this bit is set, the USART is able to wake up the MCU from Stop mode, provided that the USART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from Stop mode.

1: USART able to wake up the MCU from Stop mode. When this function is active, the clock source for the USART must be HSI or LSE (see RCC chapter)

*Note:* 1: It is recommended to set the UESM bit just before entering Stop mode and clear it on exit from Stop mode.

2: If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**Bit 0 UE:** USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the USART is kept, but all the status flags, in the USART\_ISR are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note:* 1: In order to go into low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the USART\_ISR to be set before resetting the UE bit.

2: The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

### 25.7.2 Control register 2 (USART\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:4]				ADD[3:0]				RTO EN	ABRMOD[1:0]		ABREN	MSBFI RST	DATAI NV	TXINV	RXINV
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
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				

Bits 31:28 **ADD[7:4]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with 7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. It may also be used for character detection during normal reception, Mute mode inactive (for example, end of block detection in ModBus protocol). In this case, the whole received character (8-bit) is compared to the ADD[7:0] value and CMF flag is set on match.

This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bits 27:24 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node or a character code to be recognized.

This is used in multiprocessor communication during Mute mode or Stop mode, for wakeup with address mark detection.

This bit field can only be written when reception is disabled (RE = 0) or the USART is disabled (UE=0)

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART\_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Please refer to Section 25.4: USART implementation on page 623.*

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement. (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

*Note: If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)*

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Please refer to Section 25.4: USART implementation on page 623.*

This bit field can only be written when ABREN = 0 or the USART is disabled (UE=0).

- Bit 20 **ABREN**: Auto baud rate enable  
This bit is set and cleared by software.  
0: Auto baud rate detection is disabled.  
1: Auto baud rate detection is enabled.  
*Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*
- Bit 19 **MSBFIRST**: Most significant bit first  
This bit is set and cleared by software.  
0: data is transmitted/received with data bit 0 first, following the start bit.  
1: data is transmitted/received with the MSB (bit 8/9) first, following the start bit.  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 18 **DATAINV**: Binary data inversion  
This bit is set and cleared by software.  
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)  
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 17 **TXINV**: TX pin active level inversion  
This bit is set and cleared by software.  
0: TX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd=0/mark)  
1: TX pin signal values are inverted. ( $V_{DD} = 0/\text{mark}$ , Gnd=1/idle).  
This allows the use of an external inverter on the TX line.  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 16 **RXINV**: RX pin active level inversion  
This bit is set and cleared by software.  
0: RX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd=0/mark)  
1: RX pin signal values are inverted. ( $V_{DD} = 0/\text{mark}$ , Gnd=1/idle).  
This allows the use of an external inverter on the RX line.  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 15 **SWAP**: Swap TX/RX pins  
This bit is set and cleared by software.  
0: TX/RX pins are used as defined in standard pinout  
1: The TX and RX pins functions are swapped. This allows to work in the case of a cross-wired connection to another UART.  
This bit field can only be written when the USART is disabled (UE=0).
- Bit 14 **LINEN**: LIN mode enable  
This bit is set and cleared by software.  
0: LIN mode disabled  
1: LIN mode enabled  
The LIN mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBKRQ bit in the USART\_CR1 register, and to detect LIN Sync breaks.  
This bit field can only be written when the USART is disabled (UE=0).  
*Note: If the USART does not support LIN mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

Bits 13:12 **STOP[1:0]**: STOP bits

These bits are used for programming the stop bits.

- 00: 1 stop bit
- 01: Reserved.
- 10: 2 stop bits
- 11: 1.5 stop bit

This bit field can only be written when the USART is disabled (UE=0).

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the SCLK pin.

- 0: SCLK pin disabled
- 1: SCLK pin enabled

This bit can only be written when the USART is disabled (UE=0).

*Note: If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

- 0: Steady low value on SCLK pin outside transmission window
- 1: Steady high value on SCLK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

*Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see [Figure 209](#) and [Figure 210](#))

- 0: The first clock transition is the first data capture edge
- 1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

*Note: If synchronous mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

- 0: The clock pulse of the last data bit is not output to the SCLK pin
- 1: The clock pulse of the last data bit is output to the SCLK pin

**Caution:** The last bit is the 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 can only be written when the USART is disabled (UE=0).

*Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

## Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

- 0: Interrupt is inhibited
- 1: An interrupt is generated whenever LBDF=1 in the USART\_ISR register

*Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE=0).

*Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

Bit 4 **ADDM7**: 7-bit Address Detection/4-bit Address Detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE=0)

*Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.*

Bits 3:0 Reserved, must be kept at reset value.

*Note: The 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.*

### 25.7.3 Control register 3 (USART\_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS[2:0]			SCARCNT[2:0]			Res.
									rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wakeup from Stop mode interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever WUF=1 in the USART\_ISR register

*Note:* 1. WUFIE must be set before entering in Stop mode.

2. The WUF interrupt is active only in Stop mode.

3. If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bits 21:20 **WUS[1:0]**: Wakeup from Stop mode interrupt flag selection

This bit-field specify the event which activates the WUF (Wakeup from Stop mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01:Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE.

This bit field can only be written when the USART is disabled (UE=0).

*Note:* If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bit-field specifies the number of retries in transmit and receive, in Smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE and PE bits set).

This bit field must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bit field may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

*Note:* If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

Bit 16 Reserved, must be kept at reset value.

- Bit 15 **DEP**: Driver enable polarity selection  
0: DE signal is active high.  
1: DE signal is active low.  
*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. Please refer to [Section 25.4: USART implementation on page 623](#).*  
This bit can only be written when the USART is disabled (UE=0).
- Bit 14 **DEM**: Driver enable mode  
This bit allows the user to activate the external transceiver control, through the DE signal.  
0: DE function is disabled.  
1: DE function is enabled. The DE signal is output on the RTS pin.  
*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept cleared. [Section 25.4: USART implementation on page 623](#).*  
This bit can only be written when the USART is disabled (UE=0).
- Bit 13 **DDRE**: DMA Disable on Reception Error  
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data will be transferred. (used for Smartcard mode)  
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.  
This bit can only be written when the USART is disabled (UE=0).  
*Note: The reception errors are: parity error, framing error or noise error.*
- Bit 12 **OVRDIS**: Overrun Disable  
This bit is used to disable the receive overrun detection.  
0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.  
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART\_RDR register.  
This bit can only be written when the USART is disabled (UE=0).  
*Note: This control bit allows checking the communication flow w/o reading the data.*
- Bit 11 **ONEBIT**: One sample bit method enable  
This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NF) is disabled.  
0: Three sample bit method  
1: One sample bit method  
This bit can only be written when the USART is disabled (UE=0).
- Bit 10 **CTSIE**: CTS interrupt enable  
0: Interrupt is inhibited  
1: An interrupt is generated whenever CTSIF=1 in the USART\_ISR register  
*Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

**Bit 9 CTSE:** CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while nCTS is deasserted, the transmission is postponed until nCTS is asserted.

This bit can only be written when the USART is disabled (UE=0)

*Note:* If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**Bit 8 RTSE:** RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

*Note:* If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**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

*Note:* If the USART does not support DMA, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**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

*Note:* If the USART does not support DMA, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**Bit 5 SCEN:** Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bit field can only be written when the USART is disabled (UE=0).

*Note:* If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**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 field can only be written when the USART is disabled (UE=0).

*Note:* If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**Bit 3 HDSEL:** Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

- 0: Normal mode
- 1: Low-power mode

This bit can only be written when the USART is disabled (UE=0).

*Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to Section 25.4: USART implementation on page 623.*

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

- 0: IrDA disabled
- 1: IrDA enabled

This bit can only be written when the USART is disabled (UE=0).

*Note: If IrDA mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to Section 25.4: USART implementation on page 623.*

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NF=1 in the USART\_ISR register).

- 0: Interrupt is inhibited
- 1: An interrupt is generated when FE=1 or ORE=1 or NF=1 in the USART\_ISR register.

### 25.7.4 Baud rate register (USART\_BRR)

*This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.*

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bits 15:4 BRR[15:4]

BRR[15:4] = USARTDIV[15:4]

Bits 3:0 BRR[3:0]

When OVER8 = 0, BRR [3:0] = USARTDIV [3:0]

When OVER8 = 1, BRR [2:0] = USARTDIV [3:0] shifted 1 bit to the right. BRR [3] must be kept cleared.

### 25.7.5 Guard time and prescaler register (USART\_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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, must be kept at reset value

Bits 15:8 **GT[7:0]**: Guard time value

This bit-field is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bit field can only be written when the USART is disabled (UE=0).

*Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to Section 25.4: USART implementation on page 623.*

Bits 7:0 **PSC[7:0]**: Prescaler value

– In IrDA Low-power and normal IrDA mode:

**PSC[7:0]** = IrDA Normal and Low-Power Baud Rate

Used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

– In Smartcard mode:

**PSC[4:0]**: Prescaler value

Used for programming the prescaler for dividing the USART source clock to provide the Smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bit field can only be written when the USART is disabled (UE=0).

*Note: Bits [7:5] must be kept cleared if Smartcard mode is used.*

*This bit field is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Please refer to Section 25.4: USART implementation on page 623.*

### 25.7.6 Receiver timeout register (USART\_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:24 **BLEN[7:0]**: Block Length

This bit-field gives the Block length in Smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In Smartcard mode, the Block length counter is reset when TXE=0.

This bit-field can be used also in other modes. In this case, the Block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

*Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.*

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bit-field gives the Receiver timeout value in terms of number of baud clocks.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details.

In this case, the timeout measurement is done starting from the start bit of the last received character.

*Note: This value must only be programmed once per received character.*

*Note: RTOR can be written on the fly. If the new value is lower than or equal to the counter, the RTOF flag is set.*

*This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Please refer to [Section 25.4: USART implementation on page 623](#).*

### 25.7.7 Request register (USART\_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ										
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value

Bit 4 **TXFRQ**: Transmit data flush request

Writing 1 to this bit sets the TXE flag.

This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data has not been sent due to errors (NACK) and the FE flag is active in the USART\_ISR register.

If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This allows to discard the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and sets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART\_ISR and request an automatic baud rate measurement on the next received data frame.

*Note*: : If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

### 25.7.8 Interrupt & status register (USART\_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering Stop mode.

*Note: If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART\_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wakeup from Stop mode flag

This bit is set by hardware, when a wakeup event is detected. The event is defined by the WUS bit field. It is cleared by software, writing a 1 to the WUCF in the USART\_ICR register.

An interrupt is generated if WUFIE=1 in the USART\_CR3 register.

*Note: 1. When UESM is cleared, WUF flag is also cleared.*

*2. The WUF interrupt is active only in Stop mode.*

*3. If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.*

Bit 19 **RWU**: Receiver wakeup from Mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wakeup/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART\_CR1 register.

When wakeup on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART\_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character is transmitted

1: Break character will be transmitted

- Bit 17 **CMF**: Character match flag  
This bit is set by hardware, when the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART\_ICR register.  
An interrupt is generated if CMIE=1 in the USART\_CR1 register.  
0: No Character match detected  
1: Character Match detected
- Bit 16 **BUSY**: Busy flag  
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).  
0: USART is idle (no reception)  
1: Reception on going
- Bit 15 **ABRF**: Auto baud rate flag  
This bit is set by hardware when the automatic baud rate has been set (RXNE will also be set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)  
It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART\_RQR register.  
*Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.*
- Bit 14 **ABRE**: Auto baud rate error  
This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)  
It is cleared by software, by writing 1 to the ABRRQ bit in the USART\_CR3 register.  
*Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'.*
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 **EOBF**: End of block flag  
This bit is set by hardware when a complete block has been received (for example T=1 Smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.  
An interrupt is generated if the EOBI=1 in the USART\_CR2 register.  
It is cleared by software, writing 1 to the EOBCF in the USART\_ICR register.  
0: End of Block not reached  
1: End of Block (number of characters) reached  
*Note: If Smartcard mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*

**Bit 11 RTOF:** Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART\_ICR register.

An interrupt is generated if RTOIE=1 in the USART\_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

*Note:* 1. If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

2. The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF will be set.

3. If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'.

**Bit 10 CTS:** CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the nCTS input pin.

0: nCTS line set

1: nCTS line reset

*Note:* If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

**Bit 9 CTSIF:** CTS interrupt flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART\_ICR register.

An interrupt is generated if CTSIE=1 in the USART\_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

*Note:* If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

**Bit 8 LBDF:** LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART\_ICR.

An interrupt is generated if LBDIE = 1 in the USART\_CR2 register.

0: LIN Break not detected

1: LIN break detected

*Note:* If the USART does not support LIN mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).

**Bit 7 TXE:** Transmit data register empty

This bit is set by hardware when the content of the USART\_TDR register has been transferred into the shift register. It is cleared by a write to the USART\_TDR register.

The TXE flag can also be cleared by writing 1 to the TXFRQ in the USART\_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit =1 in the USART\_CR1 register.

0: Data 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 software, writing 1 to the TCCF in the USART\_ICR register or by a write to the USART\_TDR register.

An interrupt is generated if TCIE=1 in the USART\_CR1 register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit will be set immediately.*

**Bit 5 RXNE:** Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART\_RDR register. It is cleared by a read to the USART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART\_RQR register.

An interrupt is generated if RXNEIE=1 in the USART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

**Bit 4 IDLE:** Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: 1. The IDLE bit will not be set again until the RXNE bit has been set (i.e. a new idle line occurs).*

*2. If mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.*

**Bit 3 ORE:** Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. It is cleared by a software, writing 1 to the ORECF, in the USART\_ICR register.

An interrupt is generated if RXNEIE=1 or EIE = 1 in the USART\_CR1 register.

0: No overrun error

1: Overrun error is detected

*Note: 1. When this bit is set, the RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*2. This bit is permanently forced to 0 (no overrun detection) when the OVRDIS bit is set in the USART\_CR3 register.*

Bit 2 **NF**: Noise detected flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART\_ICR register.

- 0: No noise is detected
- 1: Noise is detected

*Note:* 1. This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NF flag is set during multi buffer communication if the EIE bit is set.

2. When the line is noise-free, the NF flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to [Section 25.5.5: Tolerance of the USART receiver to clock deviation on page 637](#)).

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART\_ICR register.

In Smartcard mode, in transmission, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART\_CR1 register.

- 0: No Framing error is detected
- 1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART\_ICR register.

An interrupt is generated if PEIE = 1 in the USART\_CR1 register.

- 0: No parity error
- 1: Parity error

### 25.7.9 Interrupt flag clear register (USART\_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBOCF	RTOCF	Res.	CTSCF	LBDCF	Res.	TCCF	Res.	IDLECF	ORECF	NCF	FECF	PECF
			w	w		w	w		w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wakeup from Stop mode clear flag

Writing 1 to this bit clears the WUF flag in the USART\_ISR register.

*Note:* If the USART does not support the wakeup from Stop feature, this bit is reserved and forced by hardware to '0'.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART\_ISR register.

Bits 16:13 Reserved, must be kept at reset value.



- Bit 12 **EOBCF**: End of timeout clear flag  
Writing 1 to this bit clears the EOBF flag in the USART\_ISR register.  
*Note: If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*
- Bit 11 **RTOCF**: Receiver timeout clear flag  
Writing 1 to this bit clears the RTOF flag in the USART\_ISR register.  
*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*
- Bit 10 Reserved, must be kept at reset value.
- Bit 9 **CTSCF**: CTS clear flag  
Writing 1 to this bit clears the CTSIF flag in the USART\_ISR register.  
*Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*
- Bit 8 **LBDCF**: LIN break detection clear flag  
Writing 1 to this bit clears the LBDF flag in the USART\_ISR register.  
*Note: If LIN mode is not supported, this bit is reserved and forced by hardware to '0'. Please refer to [Section 25.4: USART implementation on page 623](#).*
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **TCCF**: Transmission complete clear flag  
Writing 1 to this bit clears the TC flag in the USART\_ISR register.
- Bit 5 Reserved, must be kept at reset value.
- Bit 4 **IDLECF**: Idle line detected clear flag  
Writing 1 to this bit clears the IDLE flag in the USART\_ISR register.
- Bit 3 **ORECF**: Overrun error clear flag  
Writing 1 to this bit clears the ORE flag in the USART\_ISR register.
- Bit 2 **NCF**: Noise detected clear flag  
Writing 1 to this bit clears the NF flag in the USART\_ISR register.
- Bit 1 **FECF**: Framing error clear flag  
Writing 1 to this bit clears the FE flag in the USART\_ISR register.
- Bit 0 **PECF**: Parity error clear flag  
Writing 1 to this bit clears the PE flag in the USART\_ISR register.

**25.7.10 Receive data register (USART\_RDR)**

Address offset: 0x24

Reset value: Undefined

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

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

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 197](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

**25.7.11 Transmit data register (USART\_TDR)**

Address offset: 0x28

Reset value: Undefined

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

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

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 197](#)).

When transmitting with the parity enabled (PCE bit set to 1 in the USART\_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE=1.*

25.7.12 USART register map

The table below gives the USART register map and reset values.

Table 90. 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_CR1	Res.	Res.	Res.	Res.	EOBIE	RTOIE	DEAT4	DEAT3	DEAT2	DEAT1	DEAT0	DEDT4	DEDT3	DEDT2	DEDT1	DEDT0	OVER8	CMIE	MME	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE		
	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		
0x04	USART_CR2	ADD[7:4]				ADD[3:0]				RTOEN	ABRMOD1	ABRMOD0	ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP[1:0]	CLKEN	CPOL	CPHA	LBCL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x08	USART_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSSEL	IRLP	IREN	EIE		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:4]										BRR[3:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]							PSC[7:0]										
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																		
0x1C	USART_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	Res.	EOBF	RTOF	CTS	CTSIF	LBDIF	TXE	TC	RXNE	IDLE	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOBCF	RTOCF	CTSCF	LBDCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]										
	Reset value																								X	X	X	X	X	X	X	X	X	X	X
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]										
	Reset value																								X	X	X	X	X	X	X	X	X	X	X

Refer to Section 2.2.2 on page 39 for the register boundary addresses.



## 26 Serial peripheral interface / inter-IC sound (SPI/I<sup>2</sup>S)

### 26.1 Introduction

The SPI/I<sup>2</sup>S interface can be used to communicate with external devices using the SPI protocol or the I<sup>2</sup>S audio protocol. SPI or I<sup>2</sup>S mode is selectable by software. SPI Motorola mode is selected by default after a device reset.

The serial peripheral interface (SPI) protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

The Inter-IC sound (I<sup>2</sup>S) protocol is also a synchronous serial communication interface. It can operate in slave or master mode with full duplex and half-duplex communication. It can address four different audio standards including the Philips I<sup>2</sup>S standard, the MSB- and LSB-justified standards and the PCM standard.

### 26.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4-bit to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to  $f_{PCLK}/2$ .
- Slave mode frequency up to  $f_{PCLK}/2$ .
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- SPI TI mode support

## 26.3 I2S main features

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode, overrun flag in reception mode (master and slave) and Frame Error Flag in reception and transmitter mode (slave only)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I<sup>2</sup>S protocols:
  - I<sup>2</sup>S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock can be output to drive an external audio component. Ratio is fixed at  $256 \times F_S$  (where  $F_S$  is the audio sampling frequency)

## 26.4 SPI/I2S implementation

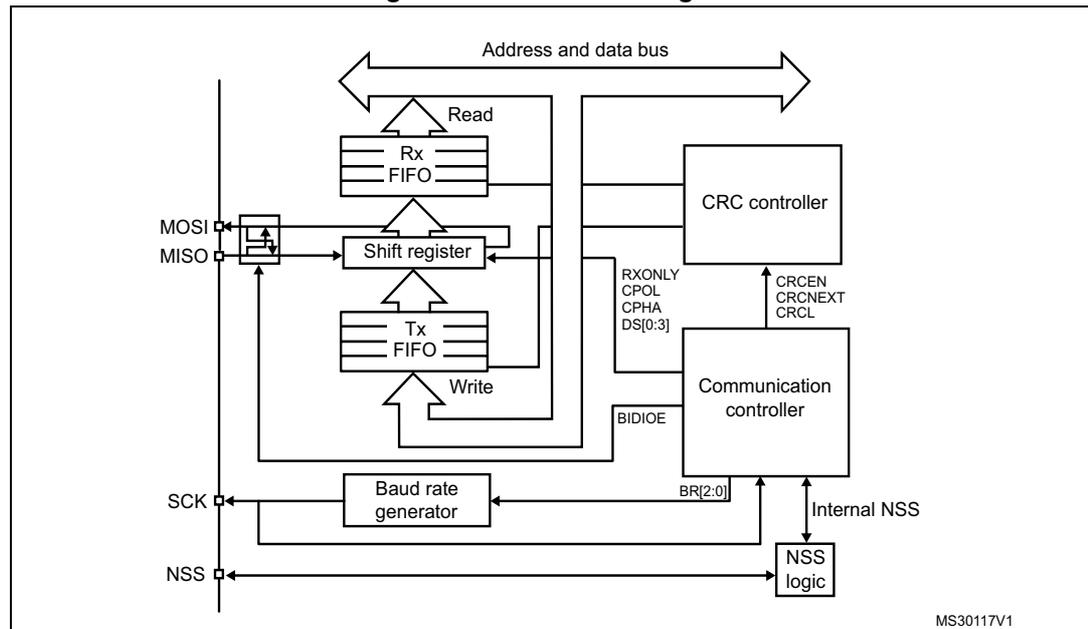
This manual describes the full set of features implemented in SPI1, SPI2 and SPI3.

## 26.5 SPI functional description

### 26.5.1 General description

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram [Figure 222](#).

Figure 222. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

See [Section 26.5.4: Slave select \(NSS\) pin management](#) for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

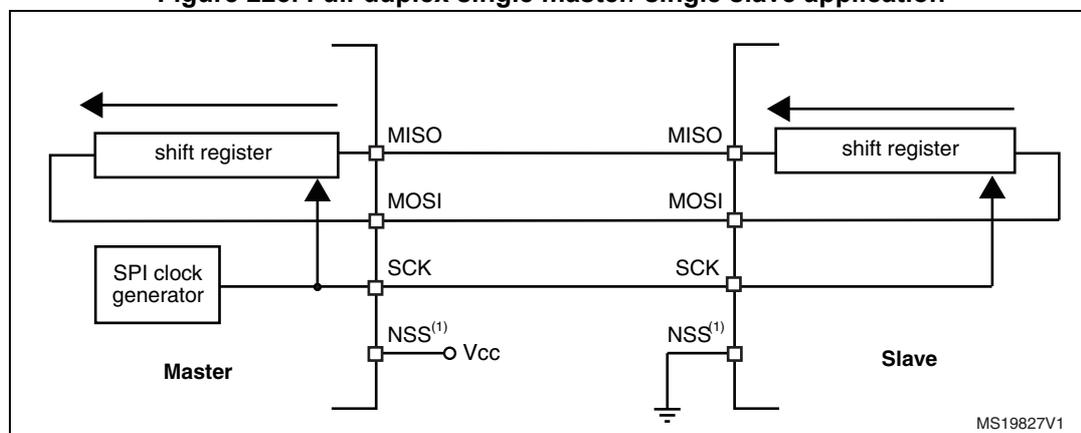
### 26.5.2 Communications between one master and one slave

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software NSS management) or 3 or 4 wires (with hardware NSS management). Communication is always initiated by the master.

#### Full-duplex communication

By default, the SPI is configured for full-duplex communication. In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During SPI communication, data is shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 223. Full-duplex single master/ single slave application

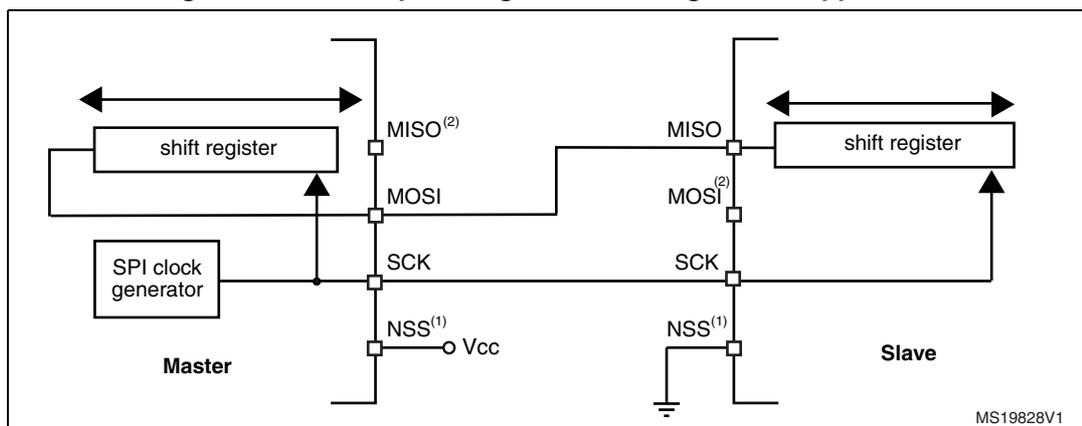


1. The NSS pin is configured as an input in this case.

#### Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx\_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx\_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 224. Half-duplex single master/ single slave application



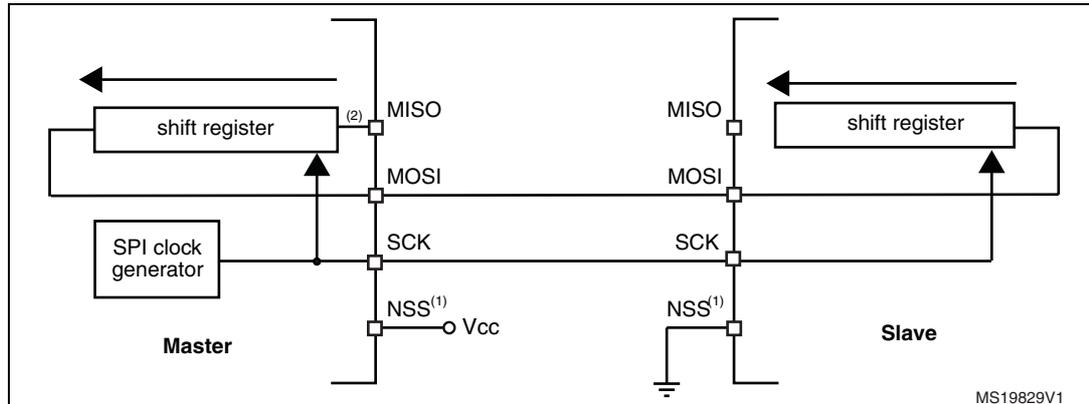
1. The NSS pin is configured as an input in this case.
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.

### Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx\_CR2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1):** The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see [26.5.4: Slave select \(NSS\) pin management](#)). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

**Figure 225. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)**



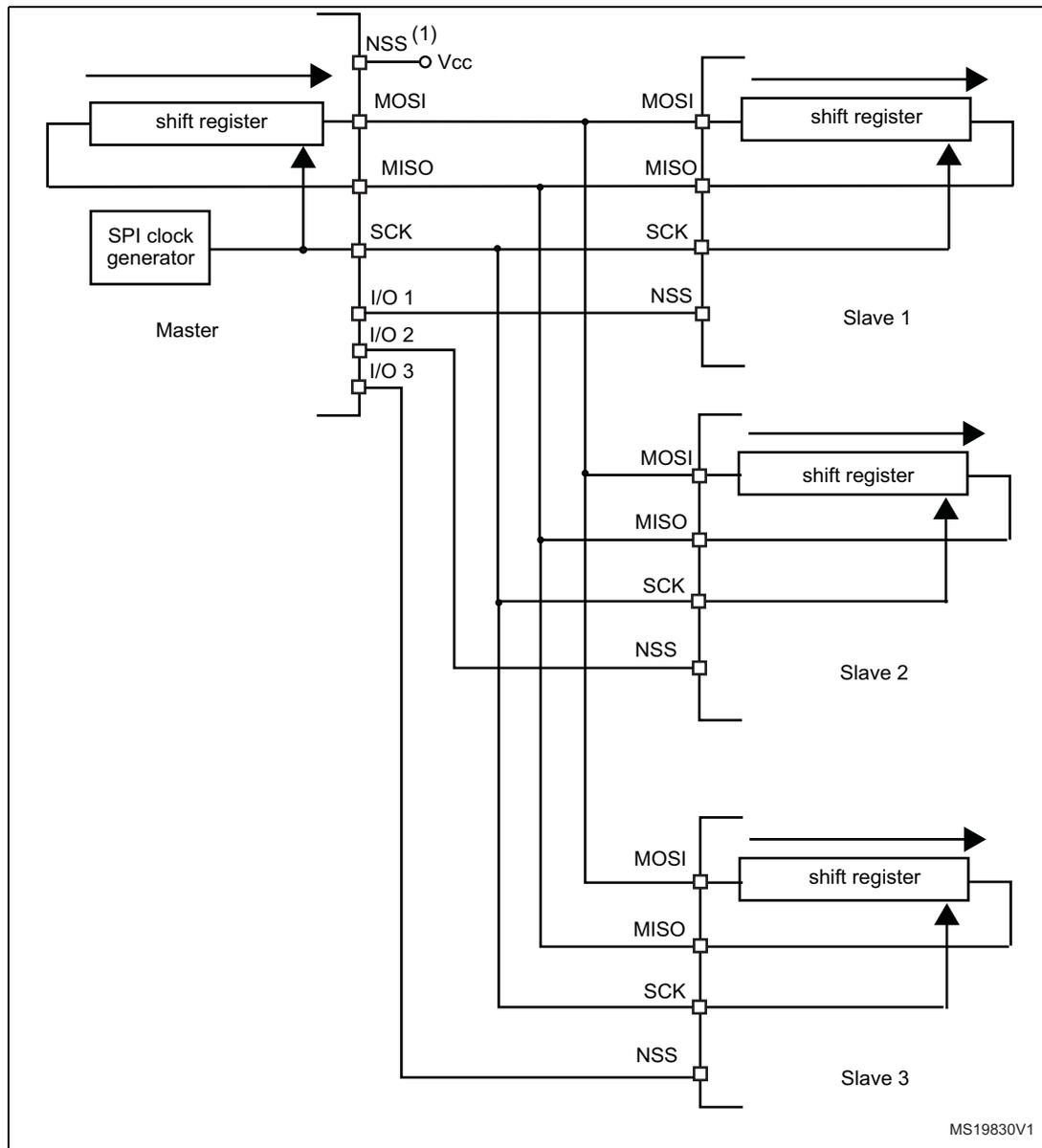
1. The NSS pin is configured as an input in this case.
2. The input information is captured in the shift register and must be ignored in standard transmit only mode (for example, OVF flag)
3. In this configuration, both the MISO pins can be used as GPIOs.

*Note:* Any simplex communication can be alternatively replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).

### 26.5.3 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see [Figure 226](#)). The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input. When this is done, a standard master and dedicated slave communication is established.

Figure 226. Master and three independent slaves



1. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see [Section 8.3.7: I/O alternate function input/output on page 142](#)).

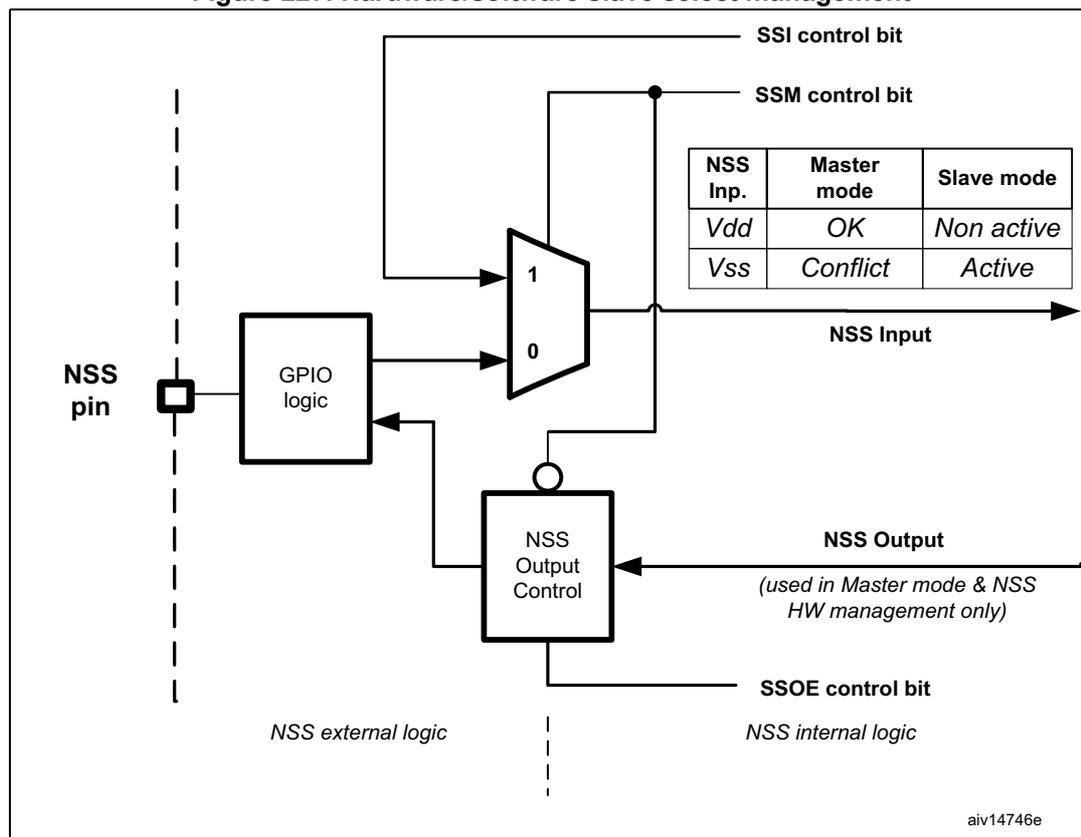
### 26.5.4 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx\_CR1 register:

- **Software NSS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in register SPIx\_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx\_CR1).
  - **NSS output enable (SSM=0,SSOE = 1):** this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0):** if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 227. Hardware/software slave select management



## 26.5.5 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

### Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx\_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

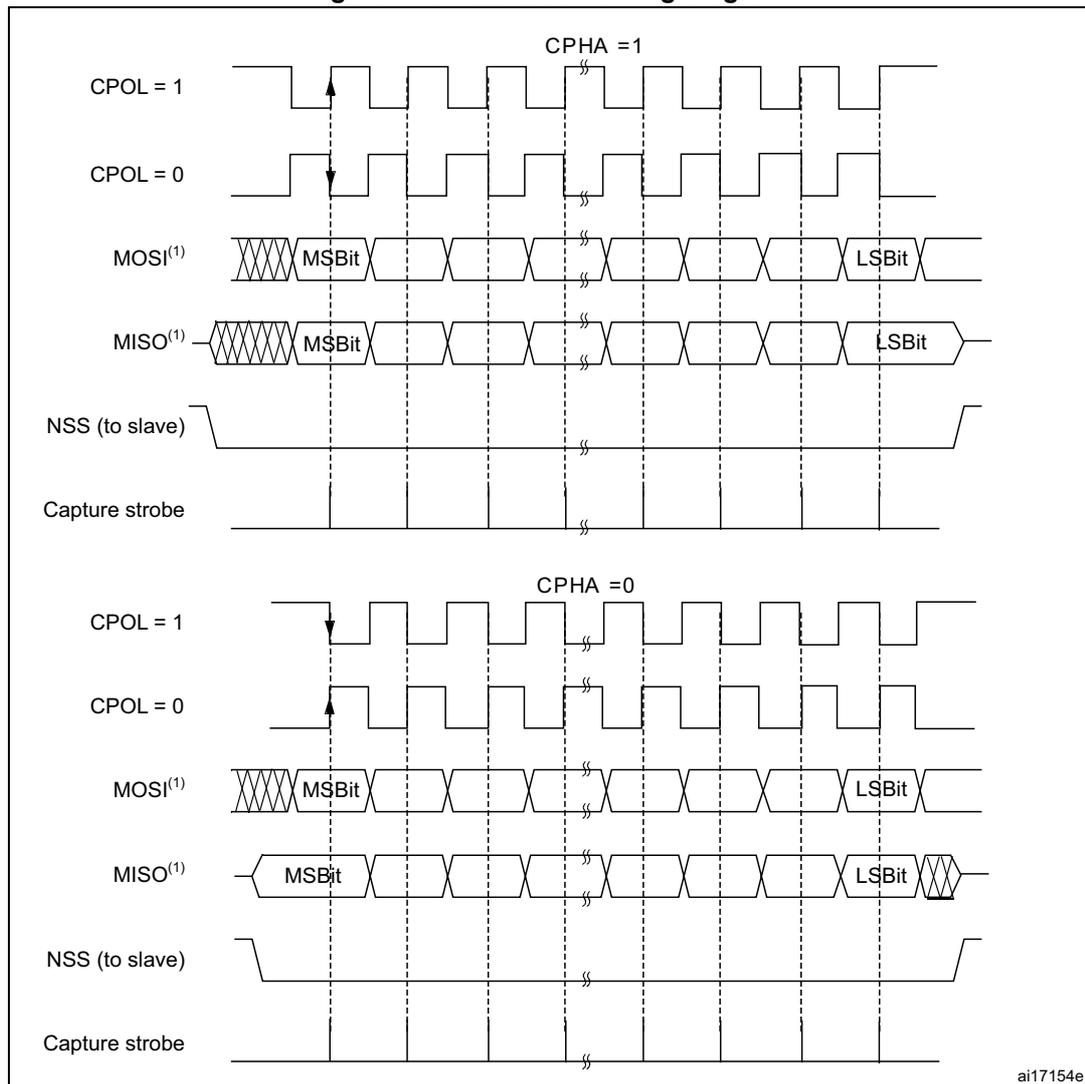
If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

*Figure 228*, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

*Note:* *Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. The idle state of SCK must correspond to the polarity selected in the SPIx\_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).*

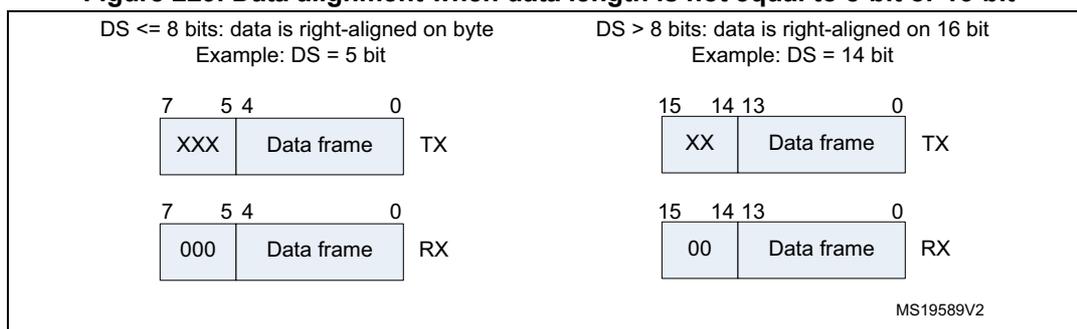
Figure 228. Data clock timing diagram



1. The order of data bits depends on LSBFIRST bit setting.

### Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFIRST bit. The data frame size is chosen by using the DS bits. It can be set from 4-bit up to 16-bit length and the setting applies for both transmission and reception. Whatever the selected data frame size, read access to the FIFO must be aligned with the FRXTH level. When the SPIx\_DR register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte) or a half-word (see [Figure 229](#)). During communication, only bits within the data frame are clocked and transferred.

**Figure 229. Data alignment when data length is not equal to 8-bit or 16-bit**

**Note:** The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 8-bit data frame size.

## 26.5.6 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated chapters. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI\_CR1 register:
  - a) Configure the serial clock baud rate using the BR[2:0] bits ([Note: 4](#)).
  - b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). ([Note: 2](#)).
  - c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE can't be set at the same time).
  - d) Configure the LSBFIRST bit to define the frame format ([Note: 2](#)).
  - e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
  - f) Configure SSM and SSI ([Note: 2 & 3](#)).
  - g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI\_CR2 register:
  - a) Configure the DS[3:0] bits to select the data length for the transfer.
  - b) Configure SSOE ([Note: 1 & 2 & 3](#)).
  - c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
  - d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CPHA and TI bits cleared in NSSP mode).
  - e) Configure the FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx\_DR register.
  - f) Initialize LDMA\_TX and LDMA\_RX bits if DMA is used in packed mode.
4. Write to SPI\_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

- Note:
- (1) Step is not required in slave mode.
  - (2) Step is not required in TI mode.
  - (3) Step is not required in NSSP mode.
  - (4) The step is not required in slave mode except slave working at TI mode

### 26.5.7 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated chapter.

### 26.5.8 Data transmission and reception procedures

#### RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see [Section 26.5.13: CRC calculation](#)).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see [Section 26.5.12: TI mode](#)).

A read access to the SPIx\_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx\_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx\_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx\_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, RXFIFO can store up to four data frames, whereas TXFIFO can only store up to three when the data frame format is not greater than 8 bits. This difference prevents possible corruption of 3x 8-bit data frames already stored in the TXFIFO when software tries to write more data in 16-bit mode into

TXFIFO. Both TXE and RXNE events can be polled or handled by interrupts. See [Figure 231](#) through [Figure 234](#).

Another way to manage the data exchange is to use DMA (see [Section 10.2: DMA main features](#)).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at [Section 26.5.9: SPI status flags](#)). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

### Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see [Section 26.5.4: Slave select \(NSS\) pin management](#)).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

### Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to

prevent some dummy byte exchange (refer to [Data packing](#) section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional "dummy" data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIIRST bits in the RCC\_APB1RSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions' streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

*Note: If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.*

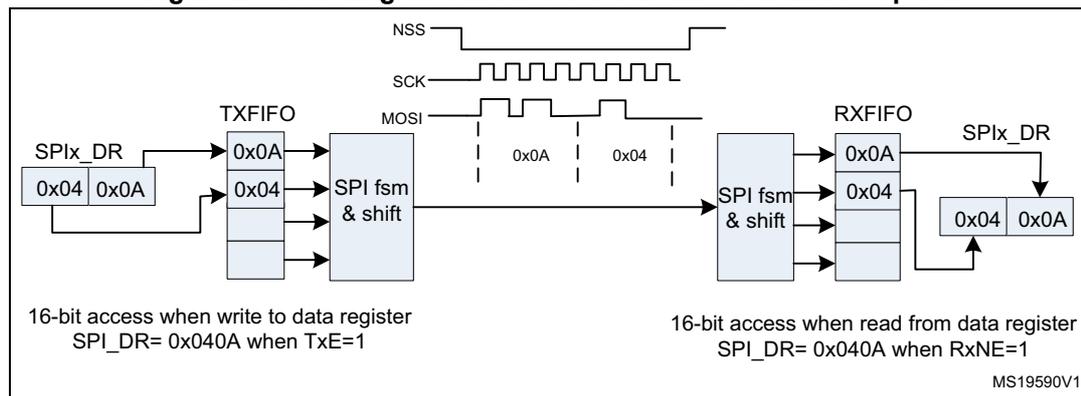
### Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx\_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. [Figure 230](#) provides an example of data packing mode sequence handling. Two data frames are sent after the single 16-bit access the SPIx\_DR register of the transmitter. This sequence can generate just one RXNE event in the receiver if the

RXFIFO threshold is set to 16 bits (FRXTH=0). The receiver then has to access both data frames by a single 16-bit read of SPIx\_DR as a response to this single RXNE event. The RxFIFO threshold setting and the following read access must be always kept aligned at the receiver side, as data can be lost if it is not in line.

A specific problem appears if an odd number of such “fit into one byte” data frames must be handled. On the transmitter side, writing the last data frame of any odd sequence with an 8-bit access to SPIx\_DR is enough. The receiver has to change the Rx\_FIFO threshold level for the last data frame received in the odd sequence of frames in order to generate the RXNE event.

**Figure 230. Packing data in FIFO for transmission and reception**



**Communication using DMA (direct memory addressing)**

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXE or RXNE enable bit in the SPIx\_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx\_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx\_DR register.

See [Figure 231](#) through [Figure 234](#).

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received is not read. When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until FTLVL[1:0]=00 and then until BSY=0.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI\_CR2 register, if DMA Rx is used.
2. Enable DMA streams for Tx and Rx in DMA registers, if the streams are used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI\_CR2 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA streams for Tx and Rx in the DMA registers, if the streams are used.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI\_CR2 register, if DMA Tx and/or DMA Rx are used.

### Packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPIx\_CR2 register) packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel. If the DMA channel PSIZE value is equal to 16-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. The DMA then automatically manages the write operations to the SPIx\_DR register.

If data packing mode is used and the number of data to transfer is not a multiple of two, the LDMA\_TX/LDMA\_RX bits must be set. The SPI then considers only one data for the transmission or reception to serve the last DMA transfer (for more details refer to [Data packing on page 699](#).)

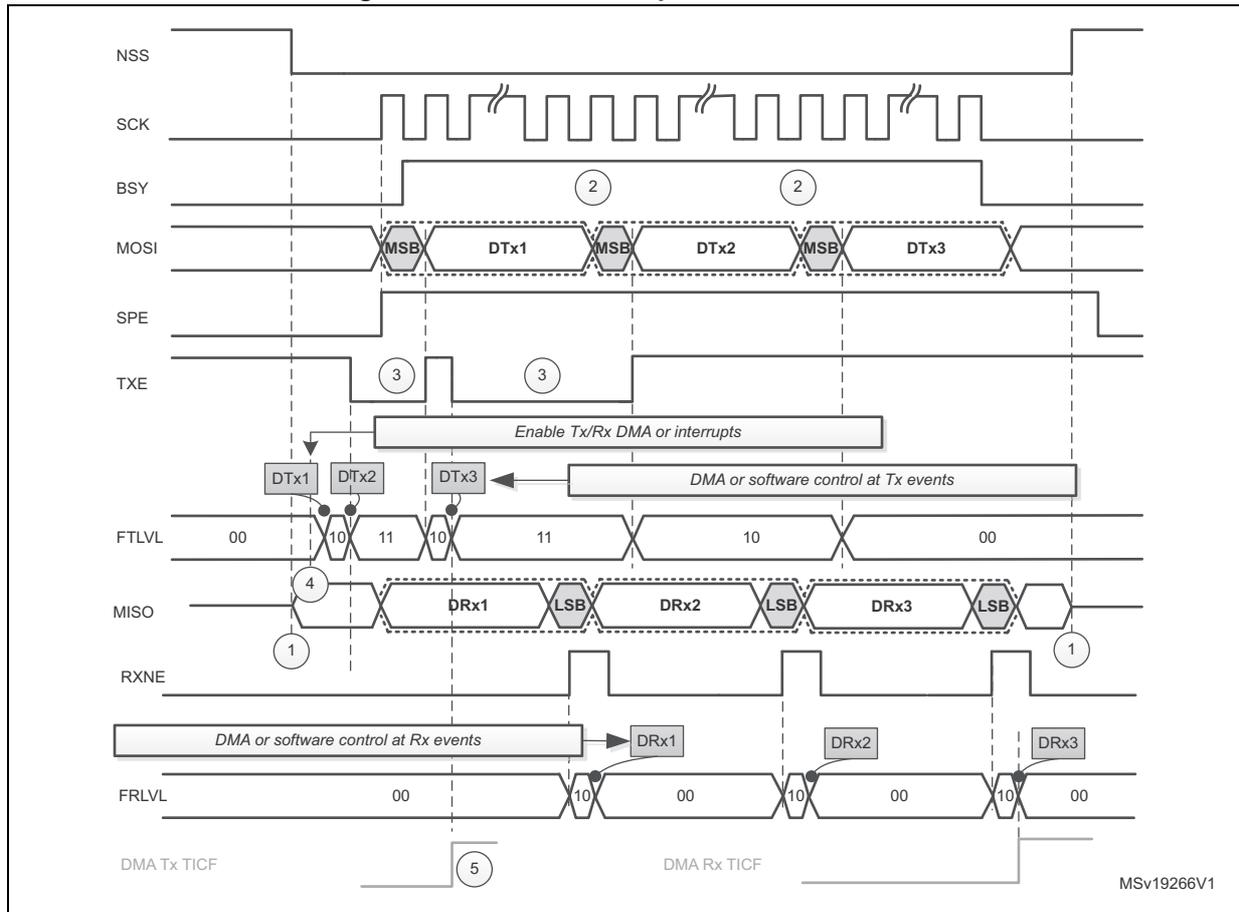
## Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by pulling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for [Figure 231 on page 703](#) through [Figure 234 on page 706](#).

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.  
At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx\_TxCRCR and SPIx\_RxCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).  
While the CRC value calculated in SPIx\_TxCRCR is simply sent out by transmitter, received CRC information is loaded into Rx FIFO and then compared with the SPIx\_RxCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of Rx FIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the Tx FIFO is  $\frac{3}{4}$  full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the Tx FIFO becomes  $\frac{1}{2}$  full. This frame is stored into Tx FIFO with an 8-bit access either by software or automatically by DMA when LDMA\_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA\_RX is set.

Figure 231. Master full duplex communication



Assumptions for master full duplex communication example:

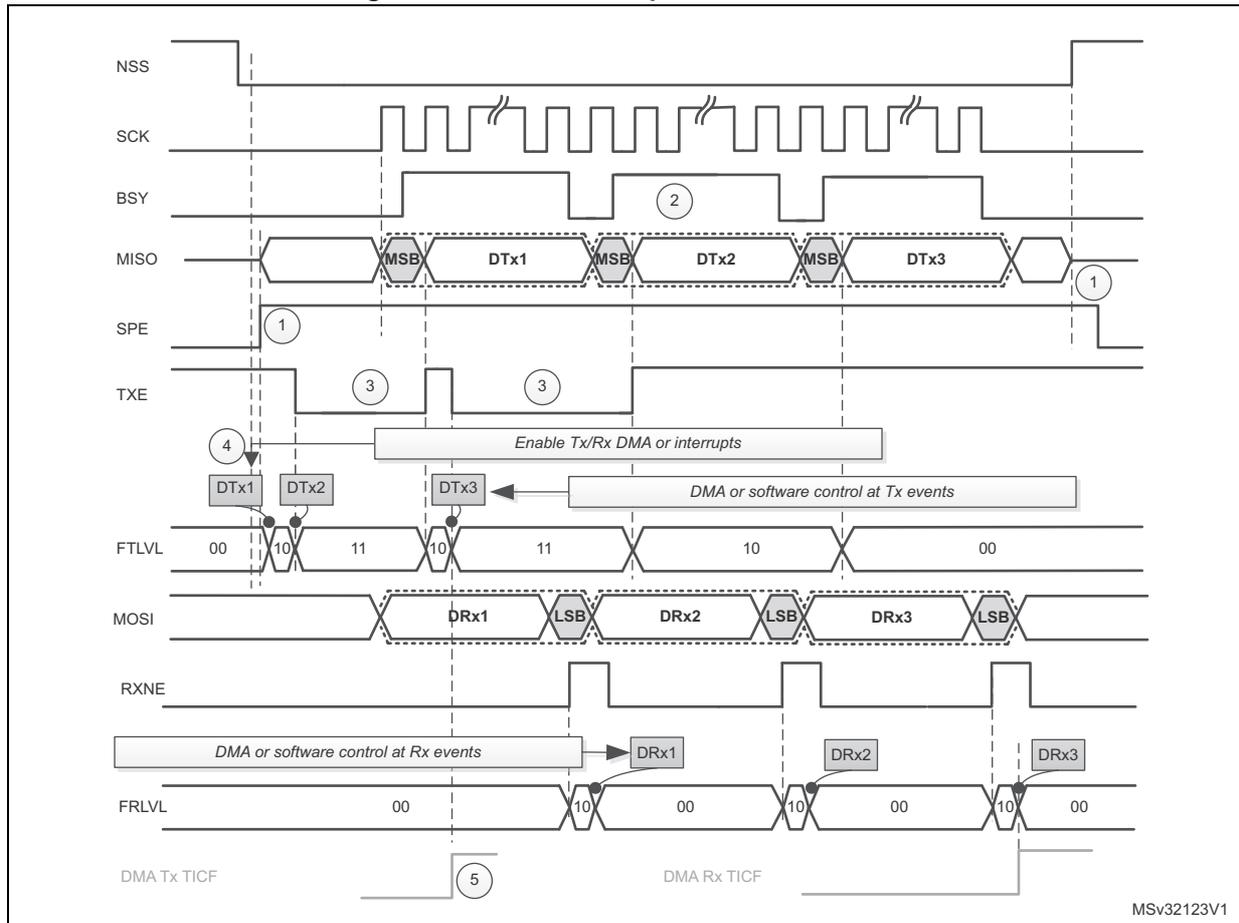
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also : [Communication diagrams on page 702](#) for details about common assumptions and notes.

Figure 232. Slave full duplex communication



Assumptions for slave full duplex communication example:

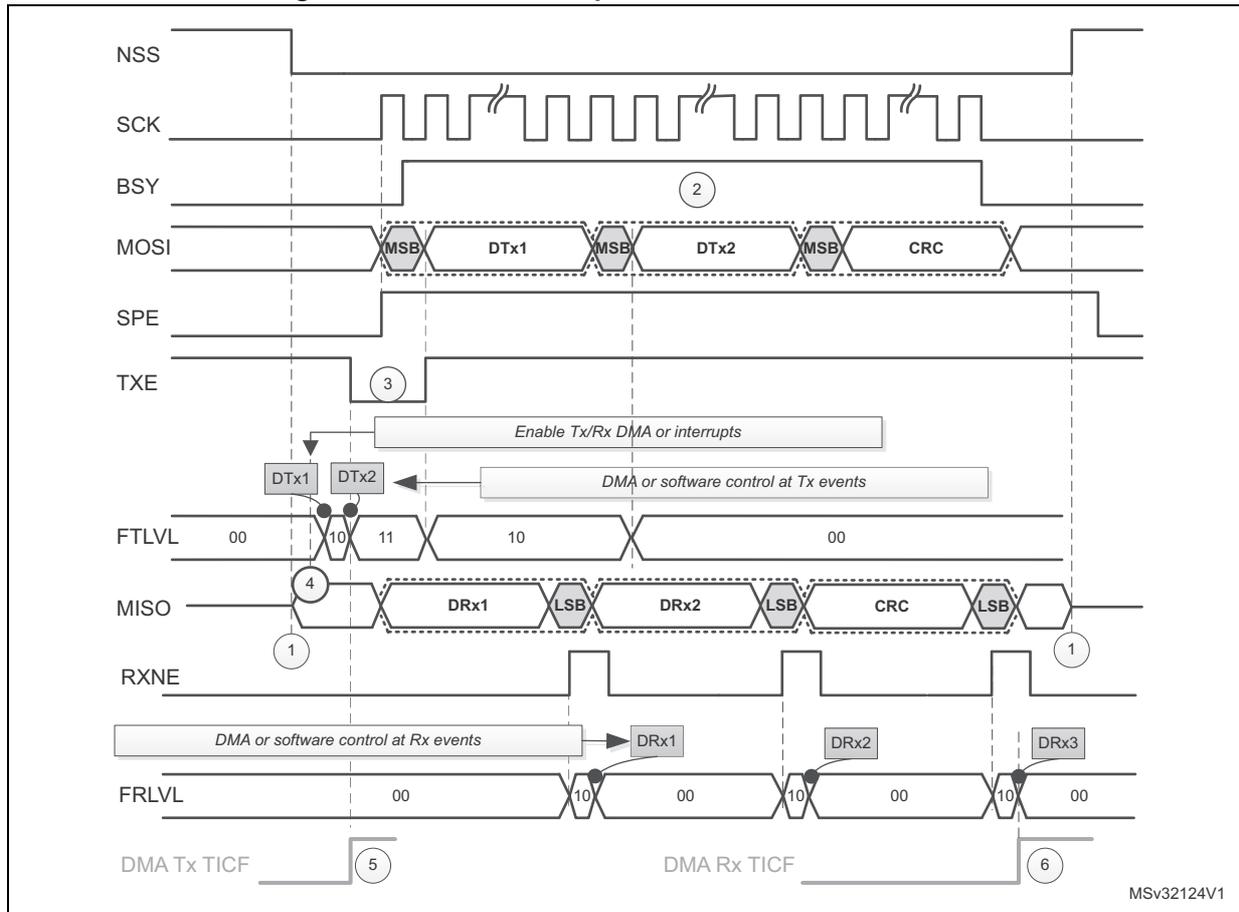
- Data size > 8 bit

If DMA is used:

- Number of Tx frames transacted by DMA is set to 3
- Number of Rx frames transacted by DMA is set to 3

See also : [Communication diagrams on page 702](#) for details about common assumptions and notes.

Figure 233. Master full duplex communication with CRC



Assumptions for master full duplex communication with CRC example:

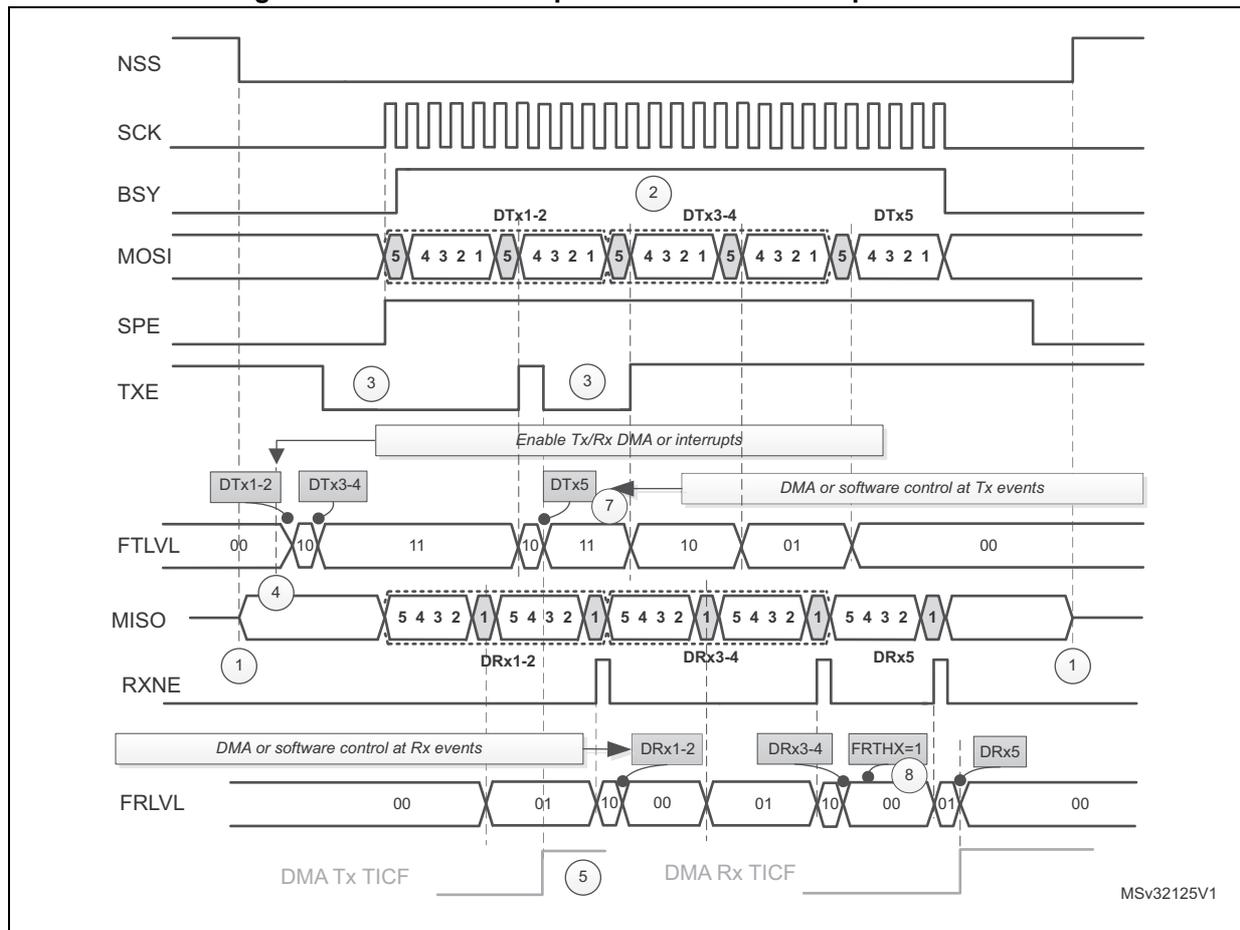
- Data size = 16 bit
- CRC enabled

If DMA is used:

- Number of Tx frames transacted by DMA is set to 2
- Number of Rx frames transacted by DMA is set to 3

See also : [Communication diagrams on page 702](#) for details about common assumptions and notes.

Figure 234. Master full duplex communication in packed mode



Assumptions for master full duplex communication in packed mode example:

- Data size = 5 bit
- Read/write FIFO is performed mostly by 16-bit access
- FRXTH=0

If DMA is used:

- Number of Tx frames to be transacted by DMA is set to 3
- Number of Rx frames to be transacted by DMA is set to 3
- PSIZE for both Tx and Rx DMA channel is set to 16-bit
- LDMA\_TX=1 and LDMA\_RX=1

See also : [Communication diagrams on page 702](#) for details about common assumptions and notes.

## 26.5.9 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

### Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx\_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

### Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx\_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx\_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

### Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

*Note:* When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.

## 26.5.10 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

### Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see [Section 26.5.13: CRC calculation](#)).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI\_DR register followed by a read access to the SPI\_SR register.

### Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx\_SR register while the MODF bit is set.
2. Then write to the SPIx\_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

### CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx\_CR1 register is set. The CRCERR flag in the SPIx\_SR register is set if the value received in the shift register does not match the receiver SPIx\_RXCRCR value. The flag is cleared by the software.

### TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx\_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

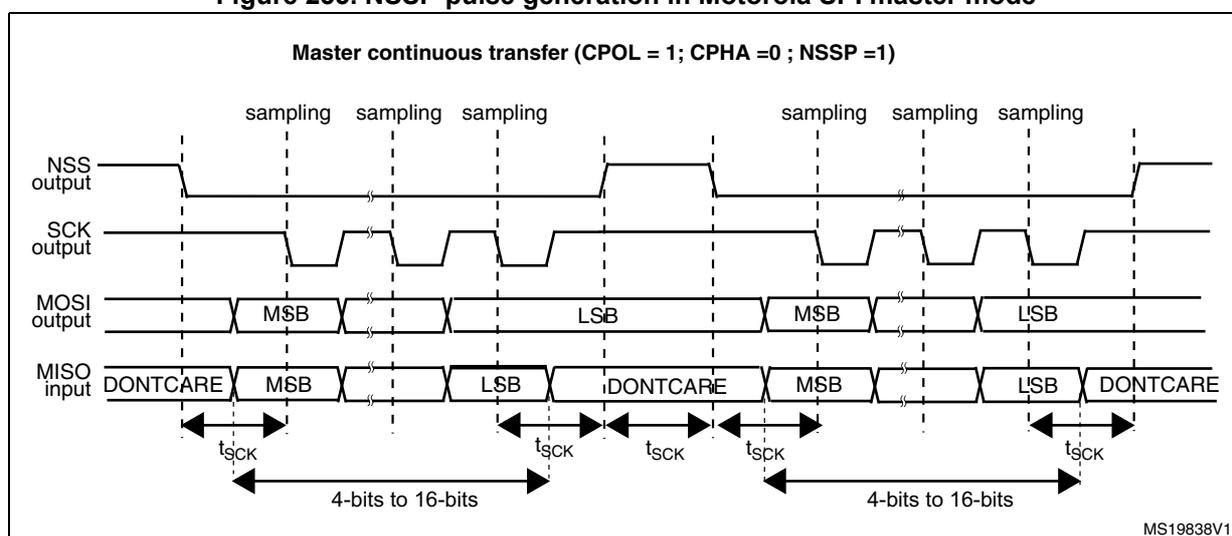
The FRE flag is cleared when SPIx\_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

### 26.5.11 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx\_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx\_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

Figure 235 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 235. NSSP pulse generation in Motorola SPI master mode



Note: Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the rising edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

### 26.5.12 TI mode

#### TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx\_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx\_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx\_CR1 and SPIx\_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see Figure 236). Any baud rate can be used, making it possible to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The delay for the MISO signal to become HiZ ( $t_{release}$ ) depends on internal resynchronization and on the

baud rate value set in through the BR[2:0] bits in the SPIx\_CR1 register. It is given by the formula:

$$\frac{t_{\text{baud\_rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud\_rate}}}{2} + 6 \times t_{\text{pclk}}$$

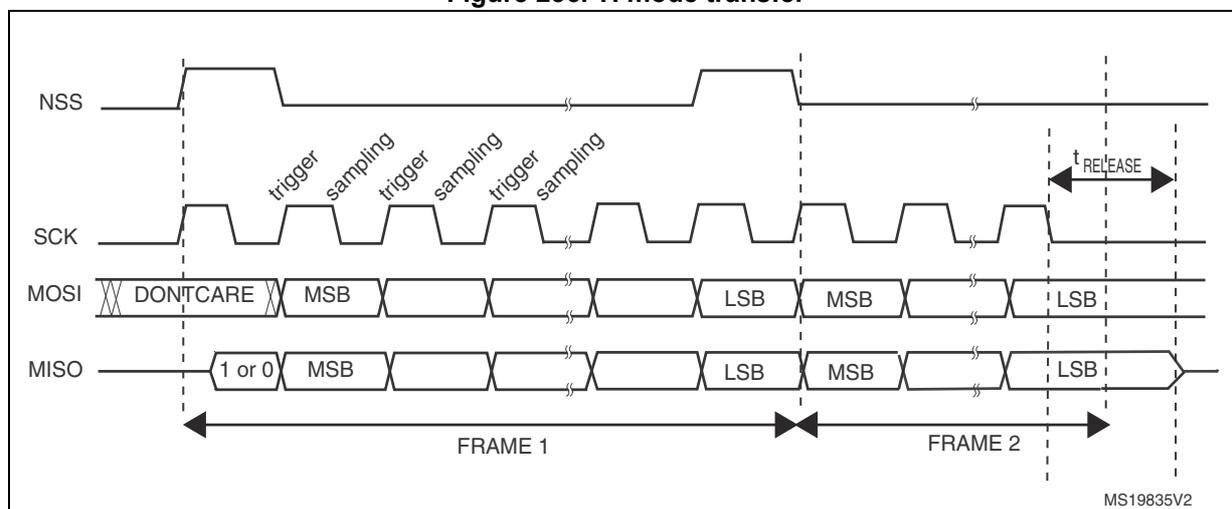
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

If the data size is equal to 4-bits or 5-bits, the master in full-duplex mode or transmit-only mode uses a protocol with one more dummy data bit added after LSB. TI NSS pulse is generated above this dummy bit clock cycle instead of the LSB in each period.

This feature is not available for Motorola SPI communications (FRF bit set to 0).

*Figure 236: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

**Figure 236. TI mode transfer**



### 26.5.13 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

#### CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx\_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx\_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

*Note: The polynomial value should only be odd. No even values are supported.*

### **CRC transfer managed by CPU**

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx\_DR register. Then CRCNEXT bit has to be set in the SPIx\_CR1 register to indicate that the CRC frame transaction will follow after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx\_RXCRC register. Software has to check the CRCERR flag in the SPIx\_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx\_DR register in order to clear the RXNE flag.

### **CRC transfer managed by DMA**

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

$$\text{DMA\_RX} = \text{Numb\_of\_data} + 2$$

In receive only mode, the DMA reception channel counter should contain only the amount of data transferred, excluding the CRC calculation. Then based on the complete transfer from DMA, all the CRC values must be read back by software from FIFO as it works as a single buffer in this mode.

At the end of the data and CRC transfers, the CRCERR flag in the SPIx\_SR register is set if corruption occurred during the transfer.

If packing mode is used, the LDMA\_RX bit needs managing if the number of data is odd.

### **Resetting the SPIx\_TXCRC and SPIx\_RXCRC values**

The SPIx\_TXCRC and SPIx\_RXCRC values are cleared automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode (not available in receive-only mode) in order to transfer data without any interruption, (several data blocks covered by intermediate CRC checking phases).

If the SPI is disabled during a communication the following sequence must be followed:

1. Disable the SPI
2. Clear the CRCEN bit
3. Enable the CRCEN bit
4. Enable the SPI

*Note: When the SPI is in slave mode, the CRC calculator is sensitive to the SCK slave input clock as soon as the CRCEN bit is set, and this is the case whatever the value of the SPE bit. In order to avoid any wrong CRC calculation, the software must enable CRC calculation only when the clock is stable (in steady state). When the SPI interface is configured as a slave, the NSS internal signal needs to be kept low between the data phase and the CRC phase.*

## 26.6 SPI interrupts

During SPI communication an interrupts can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error

Interrupts can be enabled and disabled separately.

**Table 91. SPI interrupt requests**

Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	



The I<sup>2</sup>S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I<sup>2</sup>S is configured in master mode (and when the MCKOE bit in the SPIx\_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to  $256 \times f_S$ , where  $f_S$  is the audio sampling frequency.

The I<sup>2</sup>S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I<sup>2</sup>S mode. One is linked to the clock generator configuration SPIx\_I2SPR and the other one is a generic I<sup>2</sup>S configuration register SPIx\_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx\_CR1 register and all CRC registers are not used in the I<sup>2</sup>S mode. Likewise, the SSOE bit in the SPIx\_CR2 register and the MODF and CRCERR bits in the SPIx\_SR are not used.

The I<sup>2</sup>S uses the same SPI register for data transfer (SPIx\_DR) in 16-bit wide mode.

## 26.7.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx\_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx\_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 nonsignificant bits are extended to 32 bits with 0-bits (by hardware).

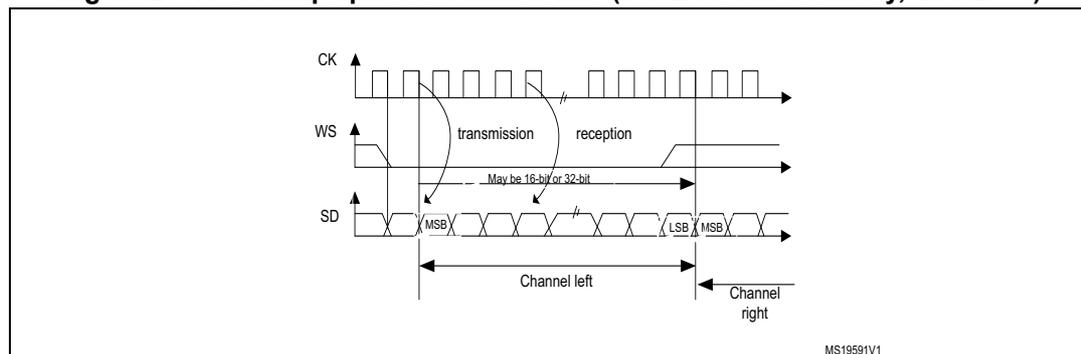
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I<sup>2</sup>S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPIx\_I2SCFGR register.

**I<sup>2</sup>S Philips standard**

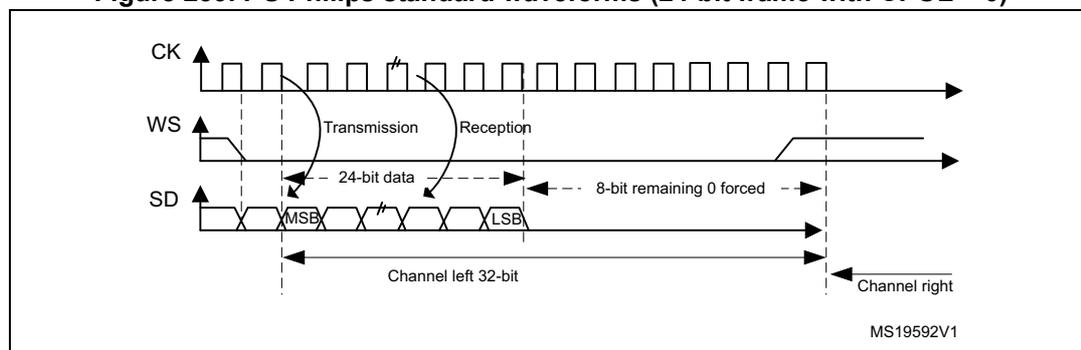
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

**Figure 238. I<sup>2</sup>S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0)**



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

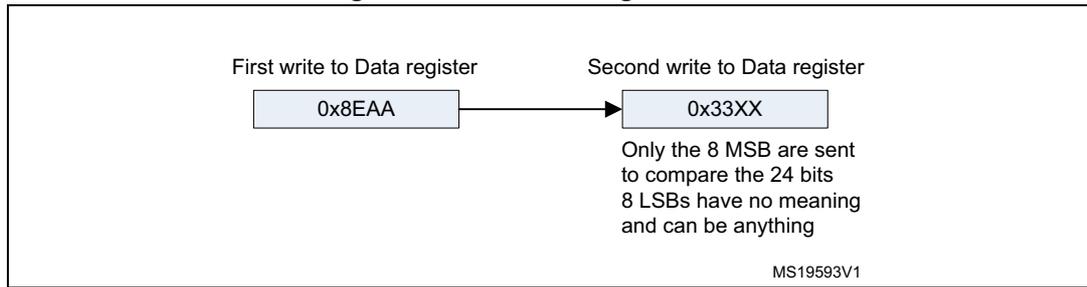
**Figure 239. I<sup>2</sup>S Philips standard waveforms (24-bit frame with CPOL = 0)**



This mode needs two write or read operations to/from the SPIx\_DR register.

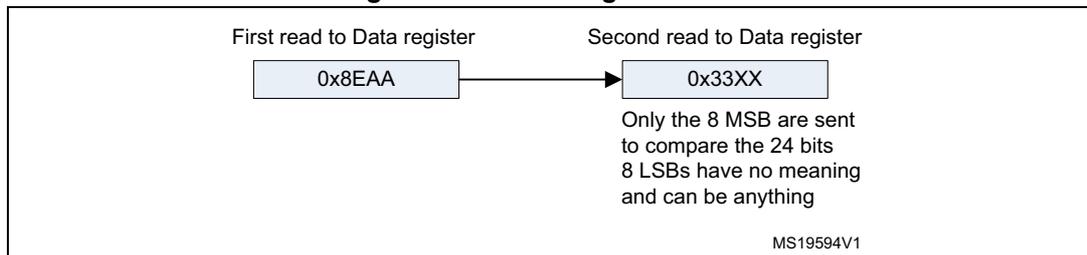
- In transmission mode:  
If 0x8EAA33 has to be sent (24-bit):

**Figure 240. Transmitting 0x8EAA33**

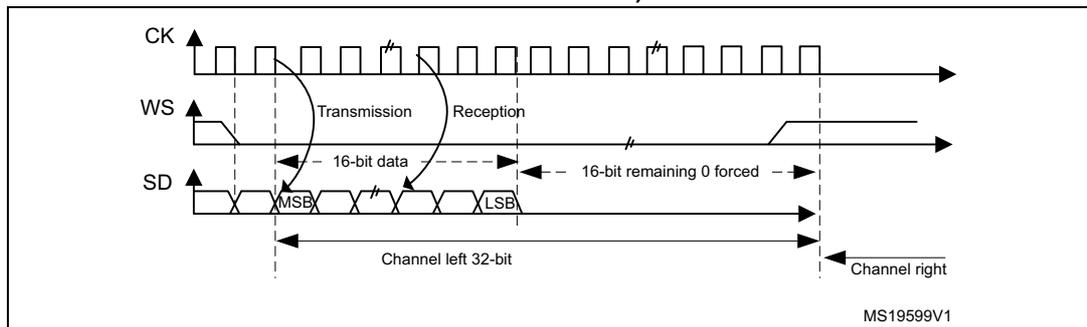


- In reception mode:  
If data 0x8EAA33 is received:

**Figure 241. Receiving 0x8EAA33**



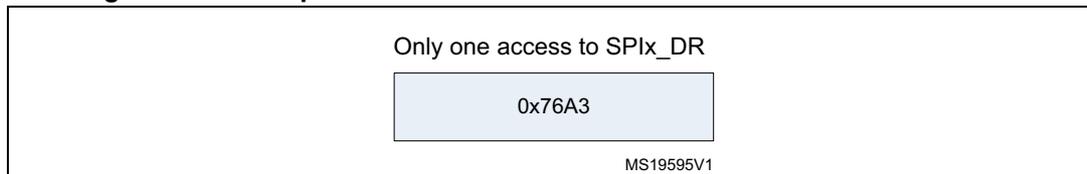
**Figure 242. I<sup>2</sup>S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)**



When 16-bit data frame extended to 32-bit channel frame is selected during the I<sup>2</sup>S configuration phase, only one access to the SPIx\_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in [Figure 243](#) is required.

**Figure 243. Example of 16-bit data frame extended to 32-bit channel frame**



For transmission, each time an MSB is written to SPIx\_DR, the TXE flag is set and its interrupt, if allowed, is generated to load the SPIx\_DR register with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

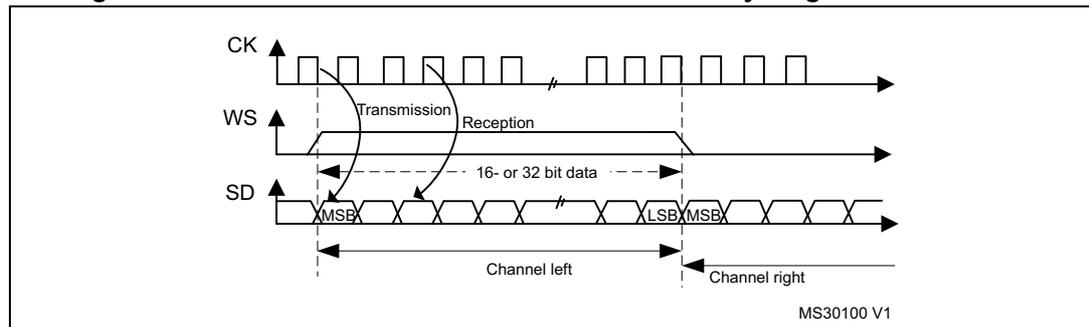
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

**MSB justified standard**

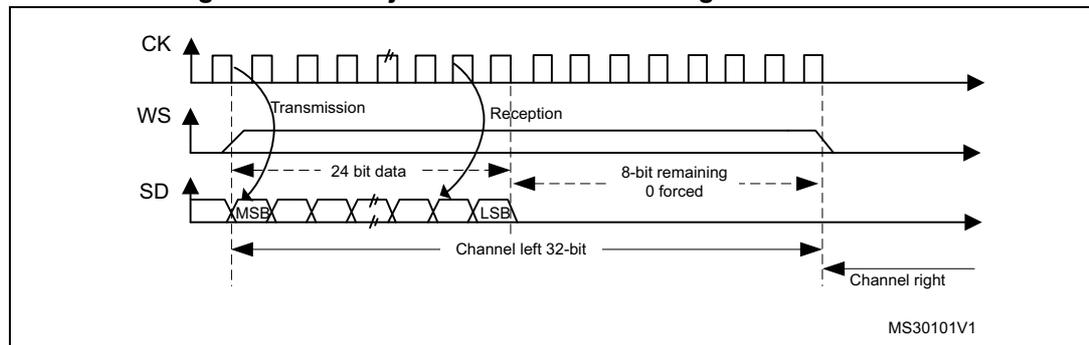
For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

**Figure 244. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0**

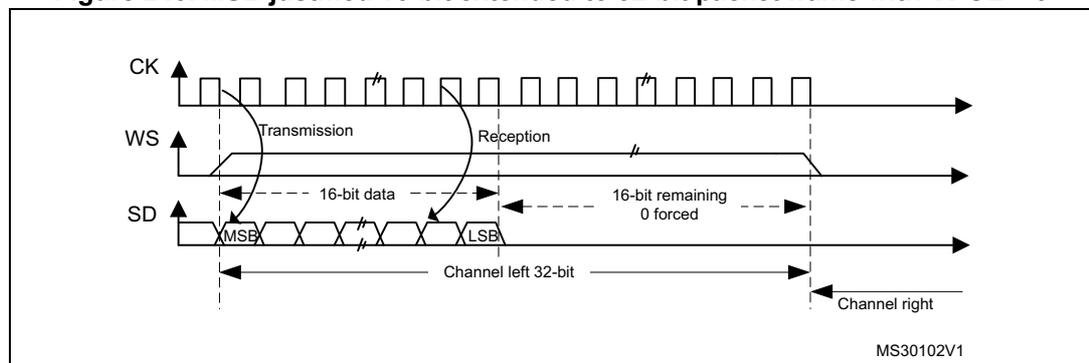


Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

**Figure 245. MSB justified 24-bit frame length with CPOL = 0**



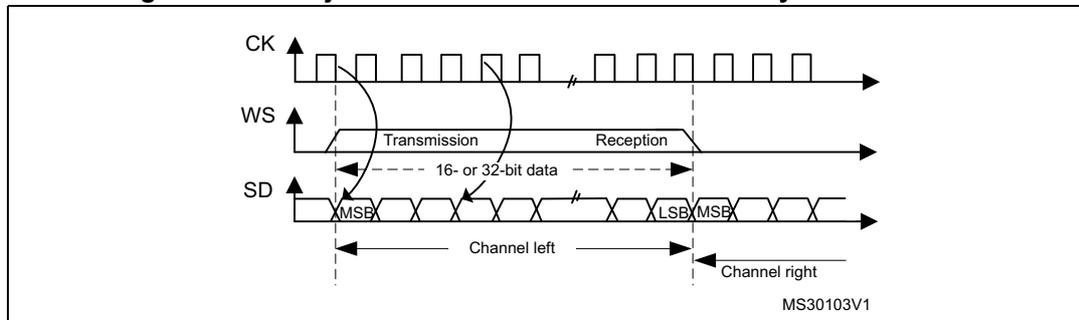
**Figure 246. MSB justified 16-bit extended to 32-bit packet frame with CPOL = 0**



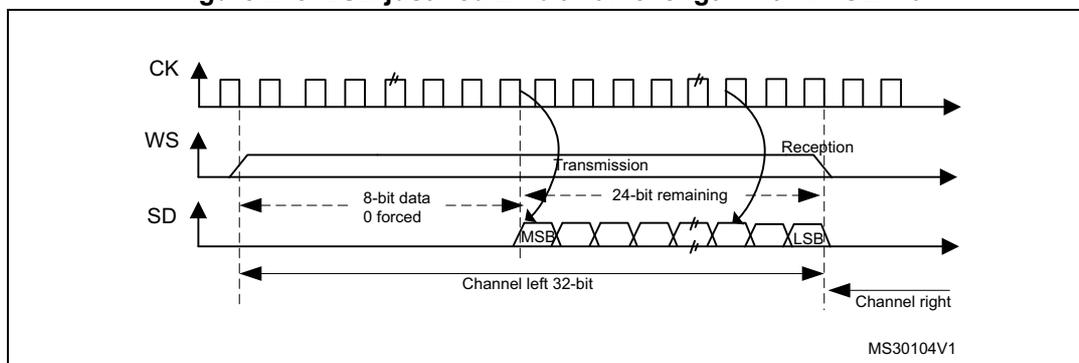
**LSB justified standard**

This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

**Figure 247. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0**

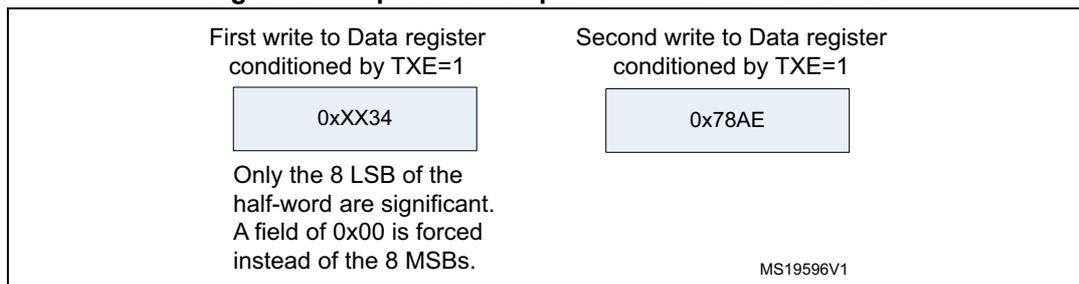


**Figure 248. LSB justified 24-bit frame length with CPOL = 0**



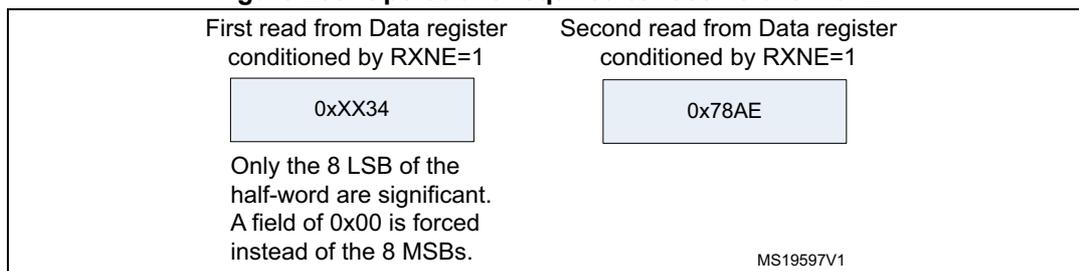
- In transmission mode:  
If data 0x3478AE have to be transmitted, two write operations to the SPIx\_DR register are required by software or by DMA. The operations are shown below.

**Figure 249. Operations required to transmit 0x3478AE**

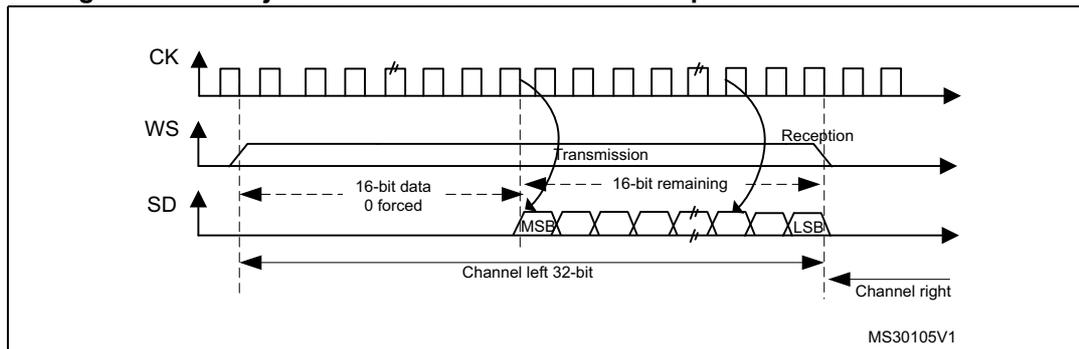


- In reception mode:  
If data 0x3478AE are received, two successive read operations from the SPIx\_DR register are required on each RXNE event.

**Figure 250. Operations required to receive 0x3478AE**



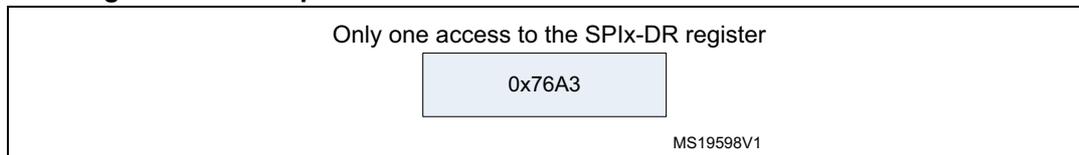
**Figure 251. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0**



When 16-bit data frame extended to 32-bit channel frame is selected during the I<sup>2</sup>S configuration phase, Only one access to the SPIx\_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in [Figure 252](#) is required.

**Figure 252. Example of 16-bit data frame extended to 32-bit channel frame**



In transmission mode, when a TXE event occurs, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). The TXE flag is set again as soon as the effective data (0x76A3) is sent on SD.

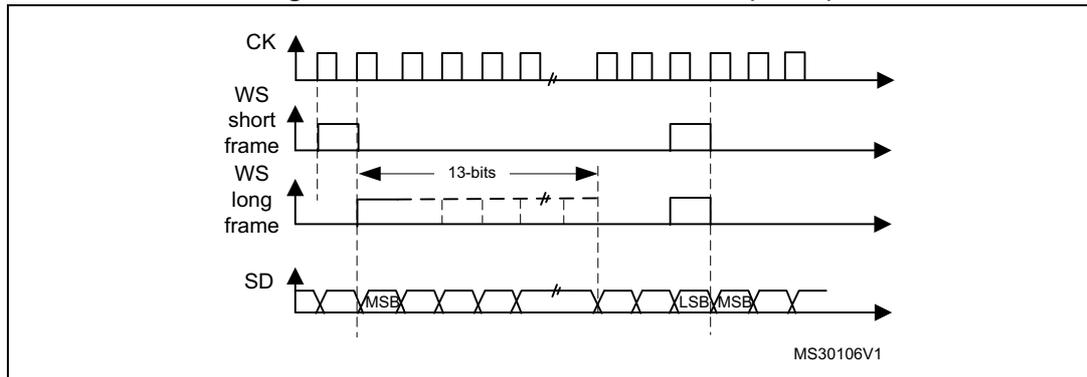
In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

**PCM standard**

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPIx\_I2SCFGR register.

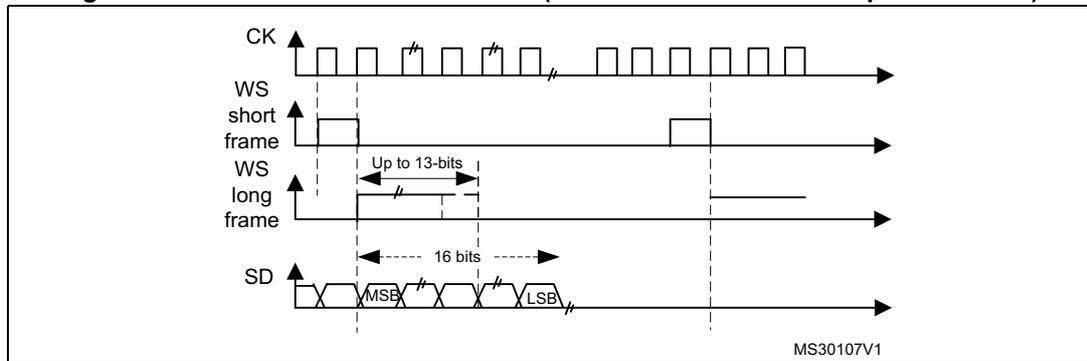
Figure 253. PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 254. PCM standard waveforms (16-bit extended to 32-bit packet frame)



**Note:** For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPIx\_I2SCFGR register) even in slave mode.

### 26.7.3 Clock generator

The I<sup>2</sup>S bitrate determines the dataflow on the I<sup>2</sup>S data line and the I<sup>2</sup>S clock signal frequency.

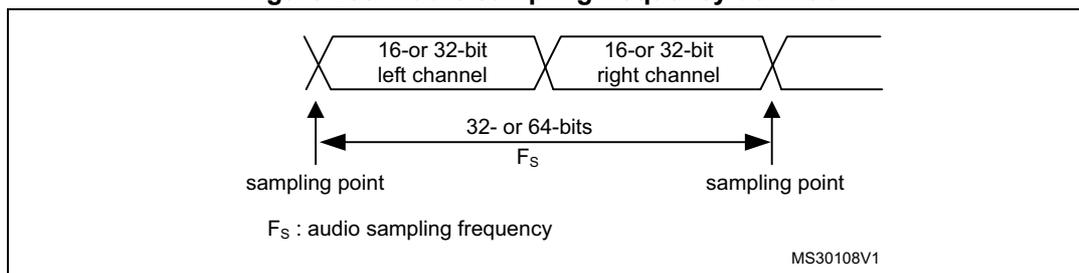
I<sup>2</sup>S bitrate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I<sup>2</sup>S bitrate is calculated as follows:

$$I^2S \text{ bitrate} = 16 \times 2 \times f_s$$

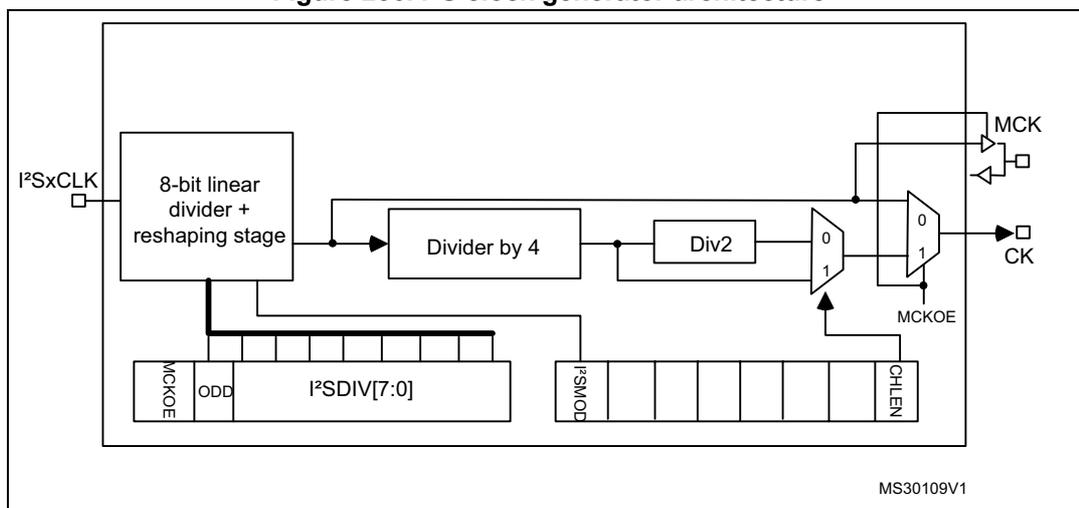
It will be: I<sup>2</sup>S bitrate = 32 × 2 × f<sub>s</sub> if the packet length is 32-bit wide.

Figure 255. Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 256. I<sup>2</sup>S clock generator architecture



1. Where x can be 2 or 3.

Figure 256 presents the communication clock architecture. The I2Sx clock is always the system clock.

The audio sampling frequency may be 192 KHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range). In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (MCKOE in the SPIx\_I2SPR register is set):

$$f_s = I2SxCLK / [(16 \cdot 2)^{((2 \cdot I2SDIV) + ODD) \cdot 8}] \text{ when the channel frame is 16-bit wide}$$

$$f_s = I2SxCLK / [(32 \cdot 2)^{((2 \cdot I2SDIV) + ODD) \cdot 4}] \text{ when the channel frame is 32-bit wide}$$

When the master clock is disabled (MCKOE bit cleared):

$$f_s = I2SxCLK / [(16 \cdot 2)^{(2 \cdot I2SDIV + ODD)}] \text{ when the channel frame is 16-bit wide}$$

$$f_s = I2SxCLK / [(32 \cdot 2)^{(2 \cdot I2SDIV + ODD)}] \text{ when the channel frame is 32-bit wide}$$

Table 93 provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

Table 92. Audio-frequency precision using standard 8 MHz HSE<sup>(1)</sup>

SYSCLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target fs (Hz)	Real fs (KHz)	Error
48	16	8	0	No	96000	93750	2.3438%
48	32	4	0	No	96000	93750	2.3438%
48	16	15	1	No	48000	48387.0968	0.8065%
48	32	8	0	No	48000	46875	2.3438%
48	16	17	0	No	44100	44117.647	0.0400%
48	32	8	1	No	44100	44117.647	0.0400%
48	16	23	1	No	32000	31914.8936	0.2660%
48	32	11	1	No	32000	32608.696	1.9022%
48	16	34	0	No	22050	22058.8235	0.0400%
48	32	17	0	No	22050	22058.8235	0.0400%
48	16	47	0	No	16000	15957.4468	0.2660%
48	32	23	1	No	16000	15957.447	0.2660%
48	16	68	0	No	11025	11029.4118	0.0400%
48	32	34	0	No	11025	11029.412	0.0400%
48	16	94	0	No	8000	7978.7234	0.2660%
48	32	47	0	No	8000	7978.7234	0.2660%
48	16	2	0	Yes	48000	46875	2.3430%
48	32	2	0	Yes	48000	46875	2.3430%
48	16	2	0	Yes	44100	46875	6.2925%
48	32	2	0	Yes	44100	46875	6.2925%
48	16	3	0	Yes	32000	31250	2.3438%
48	32	3	0	Yes	32000	31250	2.3438%
48	16	4	1	Yes	22050	20833.333	5.5178%
48	32	4	1	Yes	22050	20833.333	5.5178%
48	16	6	0	Yes	16000	15625	2.3438%
48	32	6	0	Yes	16000	15625	2.3438%
48	16	8	1	Yes	11025	11029.4118	0.0400%
48	32	8	1	Yes	11025	11029.4118	0.0400%
48	16	11	1	Yes	8000	8152.17391	1.9022%
48	32	11	1	Yes	8000	8152.17391	1.9022%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

Table 93. Audio-frequency precision using standard 8 MHz HSE<sup>(1)</sup>

SYSCLK (MHz)	I2S_DIV		I2S_ODD		MCLK	Target f <sub>s</sub> (Hz)	Real f <sub>s</sub> (KHz)		Error	
	16-bit	32-bit	16-bit	32-bit			16-bit	32-bit	16-bit	32-bit
72	11	6	1	0	No	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	No	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	No	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	No	32000	32142.86	32142.86	0.44%	0.44%
72	51	25	0	1	No	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	No	16000	15675.75	16071.43	0.27%	0.45%
72	102	51	0	0	No	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	No	8000	8007.11	7978.72	0.09%	0.27%
72	3	3	0	0	Yes	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	Yes	44100	46875	46875	6.29%	6.29%
72	9	9	0	0	Yes	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	Yes	22050	21634.61	21634.61	1.88%	1.88%
72	9	9	0	0	Yes	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	Yes	11025	10817.30	10817.30	1.88%	1.88%
72	17	17	1	1	Yes	8000	8035.71	8035.71	0.45%	0.45%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

## 26.7.4 I<sup>2</sup>S master mode

The I<sup>2</sup>S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx\_I2SPR register.

### Procedure

1. Select the I2SDIV[7:0] bits in the SPIx\_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx\_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx\_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to [Section 26.7.3: Clock generator](#)).
3. Set the I2SMOD bit in the SPIx\_I2SCFGR register to activate the I<sup>2</sup>S functions and choose the I<sup>2</sup>S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the

CHLEN bit. Select also the I<sup>2</sup>S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx\_I2SCFGR register.

4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx\_CR2 register.
5. The I2SE bit in SPIx\_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx\_I2SPR is set.

### Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Lets assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx\_CR2 register is set.

For more details about the write operations depending on the I<sup>2</sup>S standard mode selected, refer to [Section 26.7.2: Supported audio protocols](#).

To ensure a continuous audio data transmission, it is mandatory to write the SPIx\_DR register with the next data to transmit before the end of the current transmission.

To switch off the I<sup>2</sup>S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

### Reception sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in [Section 26.7.4: I2S master mode](#)), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPIx\_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPIx\_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I<sup>2</sup>S cell.

For more details about the read operations depending on the I<sup>2</sup>S standard mode selected, refer to [Section 26.7.2: Supported audio protocols](#).

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx\_CR2 register, an interrupt is generated to indicate the error.

To switch off the I<sup>2</sup>S, specific actions are required to ensure that the I<sup>2</sup>S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  - a) Wait for the second to last RXNE = 1 ( $n - 1$ )
  - b) Then wait 17 I<sup>2</sup>S clock cycles (using a software loop)
  - c) Disable the I<sup>2</sup>S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I<sup>2</sup>S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  - a) Wait for the last RXNE
  - b) Then wait 1 I<sup>2</sup>S clock cycle (using a software loop)
  - c) Disable the I<sup>2</sup>S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I<sup>2</sup>S:
  - a) Wait for the second to last RXNE = 1 ( $n - 1$ )
  - b) Then wait one I<sup>2</sup>S clock cycle (using a software loop)
  - c) Disable the I<sup>2</sup>S (I2SE = 0)

*Note:* The BSY flag is kept low during transfers.

### 26.7.5 I<sup>2</sup>S slave mode

For the slave configuration, the I<sup>2</sup>S can be configured in transmission or reception mode. The operating mode is following mainly the same rules as described for the I<sup>2</sup>S master configuration. In slave mode, there is no clock to be generated by the I<sup>2</sup>S interface. The clock and WS signals are input from the external master connected to the I<sup>2</sup>S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx\_I2SCFGR register to select I<sup>2</sup>S mode and choose the I<sup>2</sup>S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPIx\_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx\_CR2 register.
3. The I2SE bit in SPIx\_I2SCFGR register must be set.

*Note:* The I2S slave must be enabled after the external master sets the WS line at high level if the I2S protocol is selected, or at low level if the LSB or MSB-justified mode is selected.

#### Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS\_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I<sup>2</sup>S data register has to be loaded before the master initiates the communication.

For the I<sup>2</sup>S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I<sup>2</sup>S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

*Note:* The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx\_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I<sup>2</sup>S standard mode selected, refer to [Section 26.7.2: Supported audio protocols](#).

To secure a continuous audio data transmission, it is mandatory to write the SPIx\_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx\_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx\_CR2 register, an interrupt is generated when the UDR flag in the SPIx\_SR register goes high. In this case, it is mandatory to switch off the I<sup>2</sup>S and to restart a data transfer starting from the left channel.

To switch off the I<sup>2</sup>S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

### Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in [Section 26.7.5: I2S slave mode](#)), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPIx\_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPIx\_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPIx\_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from the SPIx\_DR register. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPIx\_DR register.

For more details about the read operations depending the I<sup>2</sup>S standard mode selected, refer to [Section 26.7.2: Supported audio protocols](#).

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx\_CR2 register, an interrupt is generated to indicate the error.

To switch off the I<sup>2</sup>S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

*Note:* The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

### 26.7.6 I<sup>2</sup>S status flags

Three status flags are provided for the application to fully monitor the state of the I<sup>2</sup>S bus.

#### Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I<sup>2</sup>S.

When BSY is set, it indicates that the I<sup>2</sup>S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software needs to disable the I<sup>2</sup>S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I<sup>2</sup>S is in master receiver mode.

The BSY flag is cleared:

- When a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- When the I<sup>2</sup>S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I<sup>2</sup>S clock cycle between each transfer

*Note:* Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

#### Tx buffer empty flag (TXE)

When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I<sup>2</sup>S is disabled (I2SE bit is reset).

#### RX buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPIx\_DR register is read.

#### Channel Side flag (CHSIDE)

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I<sup>2</sup>S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPIx\_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this

flag becomes meaningless and the I<sup>2</sup>S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

**When the OVR or UDR flag in the SPIx\_SR is set and the ERRIE bit in SPIx\_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx\_SR status register (once the interrupt source has been cleared).**

### 26.7.7 I<sup>2</sup>S error flags

There are three error flags for the I<sup>2</sup>S cell.

#### Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx\_DR. It is available when the I2SMOD bit in the SPIx\_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx\_CR2 register is set.

The UDR bit is cleared by a read operation on the SPIx\_SR register.

#### Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from the SPIx\_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx\_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx\_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx\_DR register followed by a read access to the SPIx\_SR register.

#### Frame error flag (FRE)

This flag can be set by hardware only if the I<sup>2</sup>S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I<sup>2</sup>S slave device:

1. Disable the I<sup>2</sup>S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in I<sup>2</sup>S mode or low for MSB- or LSB-justified or PCM modes).

Desynchronization between master and slave devices may be due to noisy environment on the SCK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

### 26.7.8 DMA features

In I<sup>2</sup>S mode, the DMA works in exactly the same way as it does in SPI mode. There is no difference except that the CRC feature is not available in I<sup>2</sup>S mode since there is no data transfer protection system.

## 26.8 I<sup>2</sup>S interrupts

[Table 94](#) provides the list of I<sup>2</sup>S interrupts.

**Table 94. I<sup>2</sup>S interrupt requests**

Interrupt event	Event flag	Enable control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	
Frame error flag	FRE	

## 26.9 SPI and I<sup>2</sup>S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI\_DR in addition by can be accessed by 8-bit access.

### 26.9.1 SPI control register 1 (SPIx\_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	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 15 **BIDIMODE**: Bidirectional data mode enable. This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

*Note: This bit is not used in I<sup>2</sup>S mode.*

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.*

*This bit is not used in I<sup>2</sup>S mode.*

Bit 13 **CRCCEN**: 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.*

*This bit is not used in I<sup>2</sup>S mode.*

Bit 12 **CRCNEXT**: Transmit CRC next

0: Next transmit value is from Tx buffer

1: Next transmit value is from Tx CRC register

*Note: This bit has to be written as soon as the last data is written in the SPIx\_DR register.*

*This bit is not used in I<sup>2</sup>S mode.*

Bit 11 **CRCL**: CRC length

This bit is set and cleared by software to select the CRC length.

0: 8-bit CRC length

1: 16-bit CRC length

*Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.*

*This bit is not used in I<sup>2</sup>S mode.*

Bit 10 **RXONLY**: Receive only mode enabled.

This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receive only mode is active. This bit is also useful in a 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)

*Note: This bit is not used in I<sup>2</sup>S mode.*

Bit 9 **SSM**: Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

- 0: Software slave management disabled
- 1: Software slave management enabled

*Note: This bit is not used in I<sup>2</sup>S mode and SPI TI mode.*

Bit 8 **SSI**: Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

*Note: This bit is not used in I<sup>2</sup>S mode and SPI TI mode.*

Bit 7 **LSBFIRST**: Frame format

- 0: data is transmitted / received with the MSB first
- 1: data is transmitted / received with the LSB first

*Note: 1. This bit should not be changed when communication is ongoing.  
2. This bit is not used in I<sup>2</sup>S mode and SPI TI mode.*

Bit 6 **SPE**: SPI enable

- 0: Peripheral disabled
- 1: Peripheral enabled

*Note: When disabling the SPI, follow the procedure described in [Procedure for disabling the SPI on page 698](#).*

*This bit is not used in I<sup>2</sup>S mode.*

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.  
This bit is not used in I<sup>2</sup>S mode.*

- Bit 2 **MSTR**: Master selection
  - 0: Slave configuration
  - 1: Master configuration

*Note: This bit should not be changed when communication is ongoing.  
This bit is not used in I<sup>2</sup>S mode.*
- 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.  
This bit is not used in I<sup>2</sup>S mode and SPI TI mode.*
- 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.  
This bit is not used in I<sup>2</sup>S mode and SPI TI mode.*

### 26.9.2 SPI control register 2 (SPIx\_CR2)

Address offset: 0x04

Reset value: 0x0700

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LDMA_TX	LDMA_RX	FRXH	DS [3:0]				TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN
	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 **LDMA\_TX**: Last DMA transfer for transmission
 

This bit is used in data packing mode, to define if the total number of data to transmit by DMA is odd or even. It has significance only if the TXDMAEN bit in the SPIx\_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx\_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx\_CR1 register).

  - 0: Number of data to transfer is even
  - 1: Number of data to transfer is odd

*Note: Refer to [Procedure for disabling the SPI on page 698](#) if the CRCEN bit is set.  
This bit is not used in I<sup>2</sup>S mode.*
- Bit 13 **LDMA\_RX**: Last DMA transfer for reception
 

This bit is used in data packing mode, to define if the total number of data to receive by DMA is odd or even. It has significance only if the RXDMAEN bit in the SPIx\_CR2 register is set and if packing mode is used (data length =< 8-bit and write access to SPIx\_DR is 16-bit wide). It has to be written when the SPI is disabled (SPE = 0 in the SPIx\_CR1 register).

  - 0: Number of data to transfer is even
  - 1: Number of data to transfer is odd

*Note: Refer to [Procedure for disabling the SPI on page 698](#) if the CRCEN bit is set.  
This bit is not used in I<sup>2</sup>S mode.*

Bit 12 **FRXTH**: FIFO reception threshold

This bit is used to set the threshold of the RXFIFO that triggers an RXNE event

- 0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit)
- 1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)

*Note: This bit is not used in I<sup>2</sup>S mode.*

Bit 11:8 **DS [3:0]**: Data size

These bits configure the data length for SPI transfers:

- 0000: Not used
- 0001: Not used
- 0010: Not used
- 0011: 4-bit
- 0100: 5-bit
- 0101: 6-bit
- 0110: 7-bit
- 0111: 8-bit
- 1000: 9-bit
- 1001: 10-bit
- 1010: 11-bit
- 1011: 12-bit
- 1100: 13-bit
- 1101: 14-bit
- 1110: 15-bit
- 1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111”(8-bit).

*Note: This bit is not used in I<sup>2</sup>S mode.*

Bit 7 **TXEIE**: Tx buffer empty interrupt enable

- 0: TXE interrupt masked
- 1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE**: RX buffer not empty interrupt enable

- 0: RXNE interrupt masked
- 1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE**: Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode, FRE at TI mode and UDR, OVR, and FRE in I<sup>2</sup>S mode).

- 0: Error interrupt is masked
- 1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

- 0: SPI Motorola mode
- 1 SPI TI mode

*Note: This bit must be written only when the SPI is disabled (SPE=0).  
This bit is not used in I<sup>2</sup>S mode.*

**Bit 3 NSSP:** NSS pulse management

This bit is used in master mode only. It allows the SPI to generate an NSS pulse between two consecutive data when doing continuous transfers. In the case of a single data transfer, it forces the NSS pin high level after the transfer.

It has no meaning if CPHA = '1', or FRF = '1'.

0: No NSS pulse

1: NSS pulse generated

*Note:* 1. This bit must be written only when the SPI is disabled (SPE=0).

2. This bit is not used in I<sup>2</sup>S mode and SPI TI mode.

**Bit 2 SSOE:** SS output enable

0: SS output is disabled in master mode and the SPI interface can work in multimaster configuration

1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multimaster environment.

*Note:* This bit is not used in I<sup>2</sup>S mode and SPI TI mode.

**Bit 1 TXDMAEN:** Tx buffer DMA enable

When this bit is set, a DMA request is generated whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

**Bit 0 RXDMAEN:** Rx buffer DMA enable

When this bit is set, a DMA request is generated whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

### 26.9.3 SPI status register (SPIx\_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTLVL[1:0]		FRLVL[2:0]		FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
			r	r	r	r	r	r	r	r	rc_w0	r	r	r	r

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:11 **FTLVL[1:0]**: FIFO Transmission Level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full (considered as FULL when the FIFO threshold is greater than 1/2)

*Note: These bits are not used in I<sup>2</sup>S mode.*

Bits 10:9 **FRLVL[1:0]**: FIFO reception level

These bits are set and cleared by hardware.

00: FIFO empty

01: 1/4 FIFO

10: 1/2 FIFO

11: FIFO full

*Note: These bits are not used in I<sup>2</sup>S mode and in SPI receive-only mode while CRC calculation is enabled.*

Bits 8 **FRE**: Frame format error

This flag is used for SPI in TI slave mode and I<sup>2</sup>S slave mode. Refer to [Section 26.5.10: SPI error flags](#) and [Section 26.7.7: I2S error flags](#).

This flag is set by hardware and reset when SPIx\_SR is read by software.

0: No frame format error

1: A frame format error occurred

Bit 7 **BSY**: Busy flag

0: SPI (or I2S) not busy

1: SPI (or I2S) is busy in communication or Tx buffer is not empty

This flag is set and cleared by hardware.

*Note: The BSY flag must be used with caution: refer to [Section 26.5.9: SPI status flags](#) and [Procedure for disabling the SPI on page 698](#).*

Bit 6 **OVR**: Overrun flag

0: No overrun occurred

1: Overrun occurred

This flag is set by hardware and reset by a software sequence. Refer to [I2S error flags on page 728](#) for the software sequence.

Bit 5 **MODF**: Mode fault

0: No mode fault occurred

1: Mode fault occurred

This flag is set by hardware and reset by a software sequence. Refer to [Section : Mode fault \(MODF\) on page 708](#) for the software sequence.

*Note: This bit is not used in I<sup>2</sup>S mode.*

Bit 4 **CRCERR**: CRC error flag

0: CRC value received matches the SPIx\_RXCRCR value

1: CRC value received does not match the SPIx\_RXCRCR value

This flag is set by hardware and cleared by software writing 0.

*Note: This bit is not used in I<sup>2</sup>S mode.*

Bit 3 **UDR**: Underrun flag

0: No underrun occurred

1: Underrun occurred

This flag is set by hardware and reset by a software sequence. Refer to [I2S error flags on page 728](#) for the software sequence.

*Note: This bit is not used in SPI mode.*

Bit 2 **CHSIDE**: Channel side

0: Channel Left has to be transmitted or has been received

1: Channel Right has to be transmitted or has been received

*Note: This bit is not used in SPI mode. It has no significance in PCM mode.*

Bit 1 **TXE**: Transmit buffer empty

0: Tx buffer not empty

1: Tx buffer empty

Bit 0 **RXNE**: Receive buffer not empty

0: Rx buffer empty

1: Rx buffer not empty

### 26.9.4 SPI data register (SPIx\_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 serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses Tx FIFO (See [Section 26.5.8: Data transmission and reception procedures](#)).*

*Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.*

### 26.9.5 SPI CRC polynomial register (SPIx\_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.

*Note: The polynomial value should be odd only. No even value is supported.*

### 26.9.6 SPI Rx CRC register (SPIx\_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 SPIx\_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx\_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx\_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx\_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note: A read to this register when the BSY Flag is set could return an incorrect value.*

*These bits are not used in I<sup>2</sup>S mode.*

### 26.9.7 SPI Tx CRC register (SPIx\_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 SPIx\_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPIx\_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (CRCL bit in the SPIx\_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (CRCL bit in the SPIx\_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note: A read to this register when the BSY flag is set could return an incorrect value.*

*These bits are not used in I<sup>2</sup>S mode.*

### 26.9.8 SPIx\_I2S configuration register (SPIx\_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved: Forced to 0 by hardware

Bit 11 **I2SMOD**: I2S mode selection

- 0: SPI mode is selected
- 1: I2S mode is selected

*Note: This bit should be configured when the SPI or I2S is disabled.*

Bit 10 **I2SE**: I2S enable

- 0: I2S peripheral is disabled
- 1: I2S peripheral is enabled

*Note: This bit is not used in SPI mode.*

Bits 9:8 **I2SCFG**: I2S configuration mode

- 00: Slave - transmit
- 01: Slave - receive
- 10: Master - transmit
- 11: Master - receive

*Note: These bits should be configured when the I2S is disabled.  
They are not used in SPI mode.*

Bit 7 **PCMSYNC**: PCM frame synchronization

- 0: Short frame synchronization
- 1: Long frame synchronization

*Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).  
It is not used in SPI mode.*

Bit 6 Reserved: forced at 0 by hardware

Bits 5:4 **I2SSTD**: I2S standard selection

- 00: I2S Philips standard.
- 01: MSB justified standard (left justified)
- 10: LSB justified standard (right justified)
- 11: PCM standard

For more details on I2S standards, refer to [Section 26.7.2 on page 714](#)

*Note: For correct operation, these bits should be configured when the I2S is disabled.  
They are not used in SPI mode.*

- Bit 3 **CKPOL**: Steady state clock polarity  
0: I<sup>2</sup>S clock steady state is low level  
1: I<sup>2</sup>S clock steady state is high level

*Note: For correct operation, this bit should be configured when the I<sup>2</sup>S is disabled.  
It is not used in SPI mode.*

- Bits 2:1 **DATLEN**: Data length to be transferred  
00: 16-bit data length  
01: 24-bit data length  
10: 32-bit data length  
11: Not allowed

*Note: For correct operation, these bits should be configured when the I<sup>2</sup>S is disabled.  
They are not used in SPI mode.*

- Bit 0 **CHLEN**: Channel length (number of bits per audio channel)  
0: 16-bit wide  
1: 32-bit wide

The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.

*Note: For correct operation, this bit should be configured when the I<sup>2</sup>S is disabled.  
It is not used in SPI mode.*

### 26.9.9 SPIx\_I2S prescaler register (SPIx\_I2SPR)

Address offset: 0x20

Reset value: 0000 0010 (0x0002)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV							
						rw	rw	rw							

Bits 15:10 Reserved: Forced to 0 by hardware

Bit 9 **MCKOE**: Master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

*Note: This bit should be configured when the I<sup>2</sup>S is disabled. It is used only when the I<sup>2</sup>S is in master mode.*

*It is not used in SPI mode.*

Bit 8 **ODD**: Odd factor for the prescaler

0: Real divider value is = I2SDIV \*2

1: Real divider value is = (I2SDIV \* 2)+1

Refer to [Section 26.7.3 on page 720](#)

*Note: This bit should be configured when the I<sup>2</sup>S is disabled. It is used only when the I<sup>2</sup>S is in master mode.*

*It is not used in SPI mode.*

Bits 7:0 **I2SDIV**: I<sup>2</sup>S linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to [Section 26.7.3 on page 720](#)

*Note: These bits should be configured when the I<sup>2</sup>S is disabled. They are used only when the I<sup>2</sup>S is in master mode.*

*They are not used in SPI mode.*

26.9.10 SPI/I2S register map

Table 95 shows the SPI/I2S register map and reset values.

Table 95. 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	SPIx_CR1	Res.	BIDIMODE	BIDIOE	CRCE	CRCEXT	CRCL	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	SPIx_CR2	Res.	LDMA TX	LDMA RX	FRXTH	DS[3:0]			TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN																	
	Reset value																		0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
0x08	SPIx_SR	Res.	FTLVL[1:0]	FRLVL[1:0]			FRE	BSY	OVR	MODF	CRCCERR	UDR	CHSIDE	TXE	RXNE																		
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	1	0	
0x0C	SPIx_DR	Res.	DR[15:0]																														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPIx_CRCPR	Res.	CRCPOLY[15:0]																														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x14	SPIx_RXCR	Res.	RxCRC[15:0]																														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPIx_TXCR	Res.	TxCRC[15:0]																														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPIx_I2SCFGR	Res.	I2SMOD	I2SE	I2SCFG	PCMSYNC	Res.	I2SSTD	CKPOL	DATLEN	CHLEN																						
	Reset value																		0	0	0	0	0	0	0	0	0						
0x20	SPIx_I2SPR	Res.	Res.	MCKOE	ODD	I2SDIV																											
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	1	0

Refer to Table 2.2.2 on page 39 for the register boundary addresses.



## 27 Touch sensing controller (TSC)

### 27.1 Introduction

The touch sensing controller provides a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode which is protected from direct touch by a dielectric (glass, plastic, ...). The capacitive variation introduced by the finger (or any conductive object) is measured using a proven implementation based on a surface charge transfer acquisition principle.

The touch sensing controller is fully supported by the STMTouch touch sensing firmware library which is free to use and allows touch sensing functionality to be implemented reliably in the end application.

### 27.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 24 capacitive sensing channels
- Up to 8 capacitive sensing channels can be acquired in parallel offering a very good response time
- Spread spectrum feature to improve system robustness in noisy environments
- Full hardware management of the charge transfer acquisition sequence
- Programmable charge transfer frequency
- Programmable sampling capacitor I/O pin
- Programmable channel I/O pin
- Programmable max count value to avoid long acquisition when a channel is faulty
- Dedicated end of acquisition and max count error flags with interrupt capability
- One sampling capacitor for up to 3 capacitive sensing channels to reduce the system components
- Compatible with proximity, touchkey, linear and rotary touch sensor implementation
- Designed to operate with STMTouch touch sensing firmware library

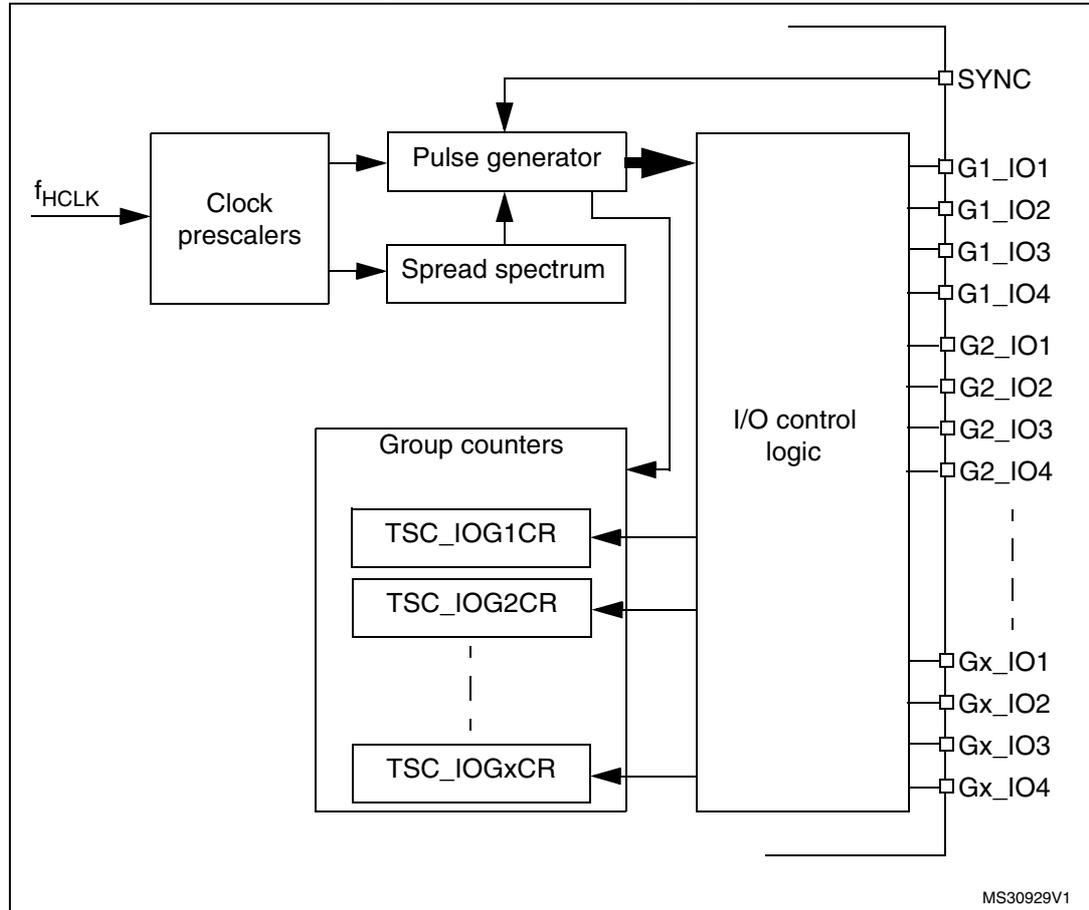
*Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.*

## 27.3 TSC functional description

### 27.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in [Figure 257: TSC block diagram](#).

Figure 257. TSC block diagram



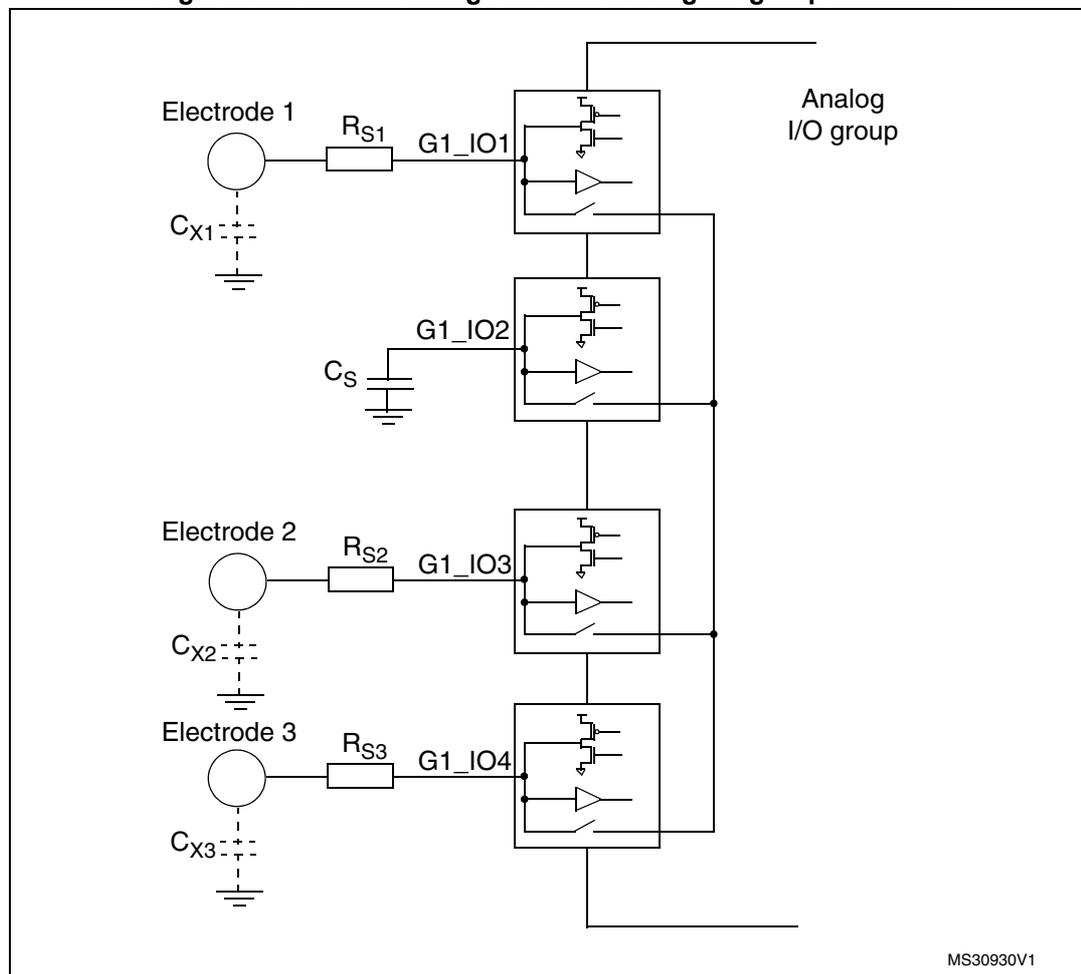
### 27.3.2 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum number of external components to operate with a single ended electrode type. This acquisition is designed around an analog I/O group which is composed of four GPIOs (see [Figure 258](#)). Several analog I/O groups are available to allow the acquisition of several capacitive sensing channels simultaneously and to support a larger number of capacitive sensing channels. Within a same analog I/O group, the acquisition of the capacitive sensing channels is sequential.

One of the GPIOs is dedicated to the sampling capacitor  $C_S$ . Only one sampling capacitor I/O per analog I/O group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels. For some specific needs (such as proximity detection), it is possible to simultaneously enable more than one channel per analog I/O group.

**Figure 258. Surface charge transfer analog I/O group structure**



MS30930V1

*Note:*  $Gx\_IOy$  where  $x$  is the analog I/O group number and  $y$  the GPIO number within the selected group.

The surface charge transfer acquisition principle consists of charging an electrode capacitance ( $C_X$ ) and transferring a part of the accumulated charge into a sampling capacitor ( $C_S$ ). This sequence is repeated until the voltage across  $C_S$  reaches a given threshold ( $V_{IH}$  in our case). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

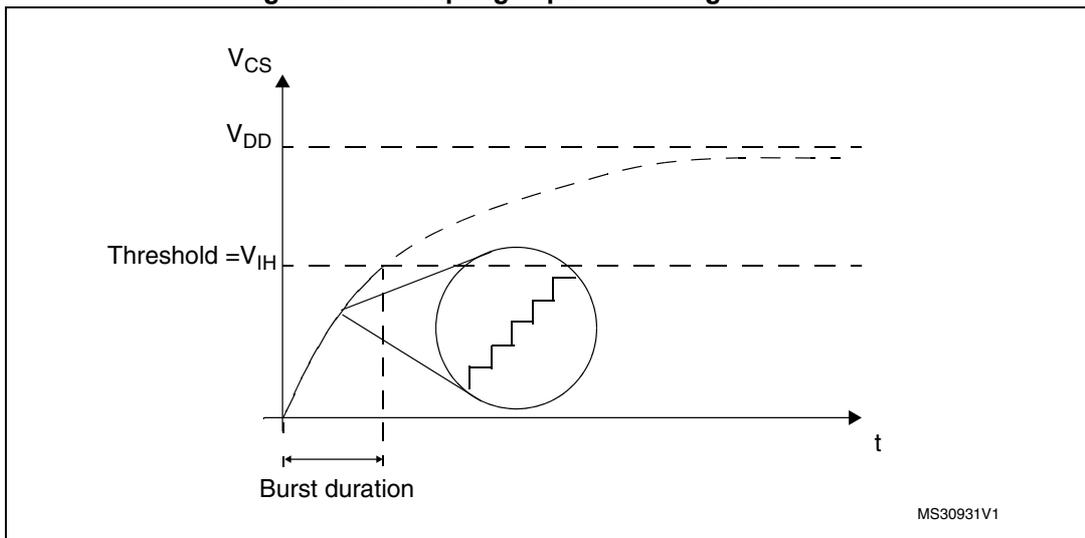
The [Table 96](#) details the charge transfer acquisition sequence of the capacitive sensing channel 1. States 3 to 7 are repeated until the voltage across  $C_S$  reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor  $R_S$  improves the ESD immunity of the solution.

**Table 96. Acquisition sequence summary**

State	G1_IO1 (electrode)	G1_IO2 (sampling)	G1_IO3 (electrode)	G1_IO4 (electrode)	State description
#1	Input floating with analog switch closed	Output open-drain low with analog switch closed	Input floating with analog switch closed		Discharge all $C_X$ and $C_S$
#2	Input floating				Dead time
#3	Output push-pull high	Input floating			Charge $C_{X1}$
#4	Input floating				Dead time
#5	Input floating with analog switch closed		Input floating		Charge transfer from $C_{X1}$ to $C_S$
#6	Input floating				Dead time
#7	Input floating				Measure $C_S$ voltage

The voltage variation over the time on the sampling capacitor  $C_S$  is detailed below:

**Figure 259. Sampling capacitor voltage variation**



### 27.3.3 Reset and clocks

The TSC clock source is the AHB clock (HCLK). Two programmable prescalers are used to generate the pulse generator and the spread spectrum internal clocks:

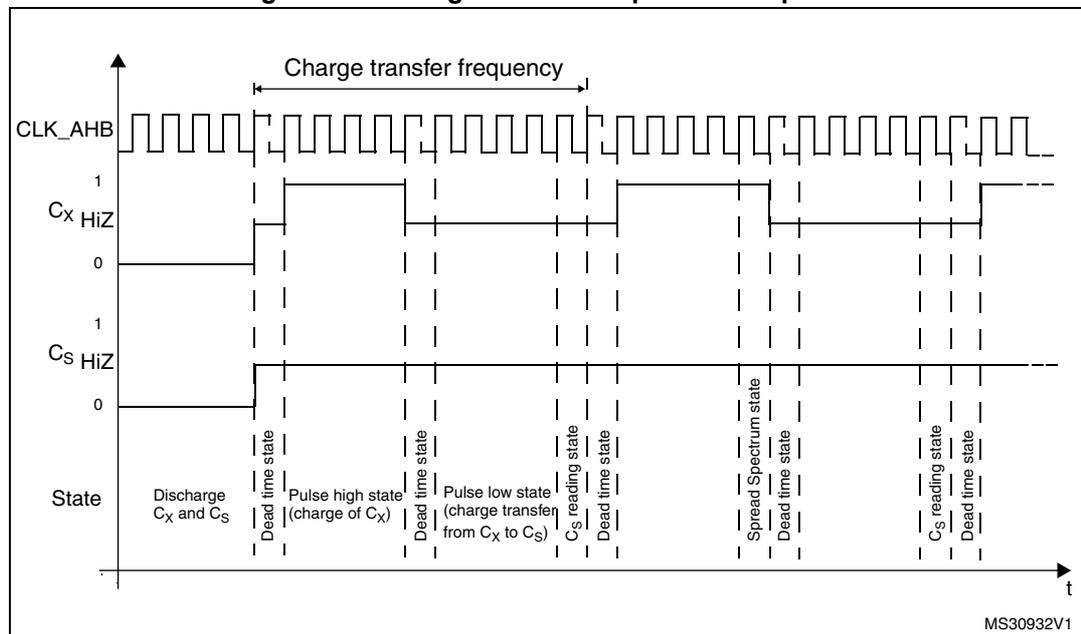
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC\_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC\_CR register

The Reset and Clock Controller (RCC) provides dedicated bits to enable the touch sensing controller clock and to reset this peripheral. For more information, please refer to [Section 7: Reset and clock control \(RCC\)](#).

### 27.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in [Figure 260](#).

**Figure 260. Charge transfer acquisition sequence**



For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of C<sub>x</sub>) and the pulse low state (transfer of charge from C<sub>x</sub> to C<sub>s</sub>) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC\_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 μs. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that C<sub>x</sub> is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 2 periods of HCLK.

At the end of the pulse high state and if the spread spectrum feature is enabled, a variable number of periods of the SSCLK clock are added.

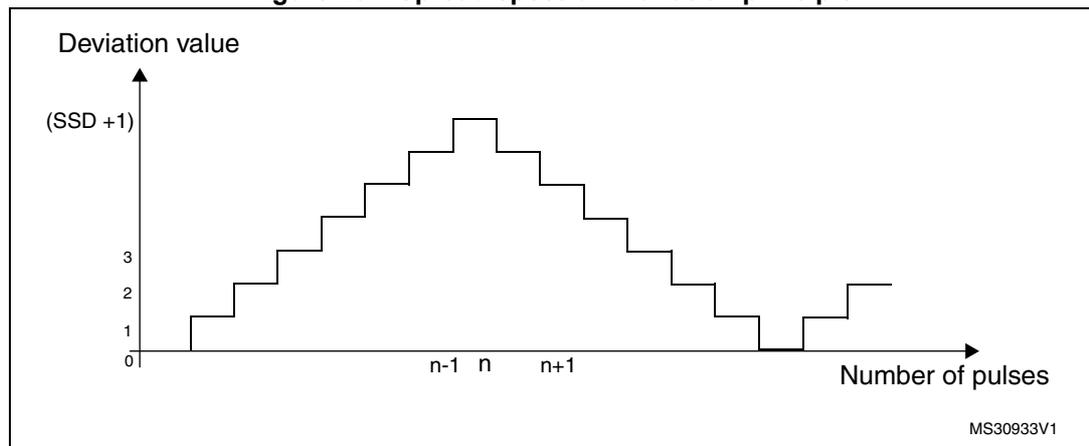
The reading of the sampling capacitor I/O, to determine if the voltage across C<sub>s</sub> has reached the given threshold, is performed at the end of the pulse low state and its duration is one period of HCLK.

### 27.3.5 Spread spectrum feature

The spread spectrum feature allows to generate a variation of the charge transfer frequency. This is done to improve the robustness of the charge transfer acquisition in noisy environments and also to reduce the induced emission. The maximum frequency variation is in the range of 10 % to 50 % of the nominal charge transfer period. For instance, for a nominal charge transfer frequency of 250 KHz (4  $\mu$ s), the typical spread spectrum deviation is 10 % (400 ns) which leads to a minimum charge transfer frequency of ~227 KHz.

In practice, the spread spectrum consists of adding a variable number of SSCLK periods to the pulse high state using the principle shown below:

Figure 261. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 97. Spread spectrum deviation versus AHB clock frequency

$f_{HCLK}$	Spread spectrum step	Maximum spread spectrum deviation
24 MHz	41.6 ns	10666.6 ns
48 MHz	20.8 ns	5333.3 ns

The spread spectrum feature can be disabled/enabled using the SSE bit in the TSC\_CR register. The frequency deviation is also configurable to accommodate the device HCLK clock frequency and the selected charge transfer frequency through the SSPSC and SSD[6:0] bits in the TSC\_CR register.

### 27.3.6 Max count error

The max count error prevents long acquisition times resulting from a faulty capacitive sensing channel. It consists of specifying a maximum count value for the analog I/O group counters. This maximum count value is specified using the MCV[2:0] bits in the TSC\_CR register. As soon as an acquisition group counter reaches this maximum value, the on-going acquisition is stopped and the end of acquisition (EOAF bit) and max count error (MCEF bit) flags are both set. An interrupt can also be generated if the corresponding end of acquisition (EOAIE bit) or/and max count error (MCEIE bit) interrupt enable bits are set.

### 27.3.7 Sampling capacitor I/O and channel I/O mode selection

To allow the GPIOs to be controlled by the touch sensing controller, the corresponding alternate function must be enabled through the standard GPIO registers and the GPIOxAFR registers.

The GPIOs modes controlled by the TSC are defined using the TSC\_IOSCR and TSC\_IOCCR register.

When there is no on-going acquisition, all the I/Os controlled by the touch sensing controller are in default state. While an acquisition is on-going, only unused I/Os (neither defined as sampling capacitor I/O nor as channel I/O) are in default state. The IODEF bit in the TSC\_CR register defines the configuration of the I/Os which are in default state. The table below summarizes the configuration of the I/O depending on its mode.

**Table 98. I/O state depending on its mode and IODEF bit value**

IODEF bit	Acquisition status	Unused I/O mode	Electrode I/O mode	Sampling capacitor I/O mode
0 (output push-pull low)	No	Output push-pull low	Output push-pull low	Output push-pull low
0 (output push-pull low)	On-going	Output push-pull low	-	-
1 (input floating)	No	Input floating	Input floating	Input floating
1 (input floating)	On-going	Input floating	-	-

#### Unused I/O mode

An unused I/O corresponds to a GPIO controlled by the TSC peripheral but not defined as an electrode I/O nor as a sampling capacitor I/O.

#### Sampling capacitor I/O mode

To allow the control of the sampling capacitor I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output open drain mode and then the corresponding Gx\_IOy bit in the TSC\_IOSCR register must be set.

Only one sampling capacitor per analog I/O group must be enabled at a time.

#### Channel I/O mode

To allow the control of the channel I/O by the TSC peripheral, the corresponding GPIO must be first set to alternate output push-pull mode and the corresponding Gx\_IOy bit in the TSC\_IOCCR register must be set.

For proximity detection where a higher equivalent electrode surface is required or to speed-up the acquisition process, it is possible to enable and simultaneously acquire several channels belonging to the same analog I/O group.

*Note:* During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOSCR or TSC\_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.

### 27.3.8 Acquisition mode

The touch sensing controller offers two acquisition modes:

- Normal acquisition mode: the acquisition starts as soon as the START bit in the TSC\_CR register is set.
- Synchronized acquisition mode: the acquisition is enabled by setting the START bit in the TSC\_CR register but only starts upon the detection of a falling edge or a rising edge and high level on the SYNC input pin. This mode is useful for synchronizing the capacitive sensing channels acquisition with an external signal without additional CPU load.

The GxE bits in the TSC\_I OGCSR registers specify which analog I/O groups are enabled (corresponding counter is counting). The  $C_S$  voltage of a disabled analog I/O group is not monitored and this group does not participate in the triggering of the end of acquisition flag. However, if the disabled analog I/O group contains some channels, they will be pulsed.

When the  $C_S$  voltage of an enabled analog I/O group reaches the given threshold, the corresponding GxS bit of the TSC\_I OGCSR register is set. When the acquisition of all enabled analog I/O groups is complete (all GxS bits of all enabled analog I/O groups are set), the EOAF flag in the TSC\_ISR register is set. An interrupt request is generated if the EOAI E bit in the TSC\_I ER register is set.

In the case that a max count error is detected, the on-going acquisition is stopped and both the EOAF and MCEF flags in the TSC\_ISR register are set. Interrupt requests can be generated for both events if the corresponding bits (EOAI E and MCEI E bits of the TSC\_I ER register) are set. Note that when the max count error is detected the remaining GxS bits in the enabled analog I/O groups are not set.

To clear the interrupt flags, the corresponding EOAI C and MCEI C bits in the TSC\_I CR register must be set.

The analog I/O group counters are cleared when a new acquisition is started. They are updated with the number of charge transfer cycles generated on the corresponding channel(s) upon the completion of the acquisition.

### 27.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller also allows to take the control of the Schmitt trigger hysteresis and analog switch of each Gx\_I O<sub>y</sub>. This control is available whatever the I/O control mode is (controlled by standard GPIO registers or other peripherals, ...) assuming that the touch sensing controller is enabled. This may be useful to perform a different acquisition sequence or for other purposes.

In order to improve the system immunity, the Schmitt trigger hysteresis of the GPIOs controlled by the TSC must be disabled by resetting the corresponding Gx\_I O<sub>y</sub> bit in the TSC\_I OHCR register.

### 27.3.10 Capacitive sensing GPIOs

The table below provides an overview of the capacitive sensing GPIOs.

**Table 99. Capacitive sensing GPIOs available on STM32F378xx devices**

Group	Capacitive sensing signal name	Pin name	Group	Capacitive sensing signal name	Pin name
1	TSC_G1_IO1	PA0	5	TSC_G5_IO1	PB3
	TSC_G1_IO2	PA1		TSC_G5_IO2	PB4
	TSC_G1_IO3	PA2		TSC_G5_IO3	PB6
	TSC_G1_IO4	PA3		TSC_G5_IO4	PB7
2	TSC_G2_IO1	PA4	6	TSC_G6_IO1	PB14
	TSC_G2_IO2	PA5		TSC_G6_IO2	PB15
	TSC_G2_IO3	PA6		TSC_G6_IO3	PD8
	TSC_G2_IO4	PA7		TSC_G6_IO4	PD9
3	TSC_G3_IO1	PC4	7	TSC_G7_IO1	PE2
	TSC_G3_IO2	PC5		TSC_G7_IO2	PE3
	TSC_G3_IO3	PB0		TSC_G7_IO3	PE4
	TSC_G3_IO4	PB1		TSC_G7_IO4	PE5
4	TSC_G4_IO1	PA9	8	TSC_G8_IO1	PD12
	TSC_G4_IO2	PA10		TSC_G8_IO2	PD13
	TSC_G4_IO3	PA13		TSC_G8_IO3	PD14
	TSC_G4_IO4	PA14		TSC_G8_IO4	PD15

## 27.4 TSC low-power modes

**Table 100. Effect of low-power modes on TSC**

Mode	Description
Sleep	No effect TSC interrupts cause the device to exit Sleep mode.
Stop	TSC registers are frozen
Standby	The TSC stops its operation until the Stop or Standby mode is exited.

## 27.5 TSC interrupts

Table 101. Interrupt control bits

Interrupt event	Enable control bit	Event flag	Clear flag bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
End of acquisition	EOAIE	EOAIF	EOAIC	yes	no	no
Max count error	MCEIE	MCEIF	MCEIC	yes	no	no

## 27.6 TSC registers

Refer to [Section 1.1 on page 36](#) of the reference manual for a list of abbreviations used in register descriptions.

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

### 27.6.1 TSC control register (TSC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
CTPH[3:0]				CTPL[3:0]				SSD[6:0]								SSE
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV[2:0]			IODEF	SYNC POL	AM	START	TSCE	
r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	

Bits 31:28 **CTPH[3:0]**: Charge transfer pulse high

These bits are set and cleared by software. They define the duration of the high state of the charge transfer pulse (charge of  $C_X$ ).

0000:  $1 \times t_{PGCLK}$   
 0001:  $2 \times t_{PGCLK}$

...

1111:  $16 \times t_{PGCLK}$

*Note: These bits must not be modified when an acquisition is on-going.*

Bits 27:24 **CTPL[3:0]**: Charge transfer pulse low

These bits are set and cleared by software. They define the duration of the low state of the charge transfer pulse (transfer of charge from  $C_X$  to  $C_S$ ).

0000:  $1 \times t_{PGCLK}$   
 0001:  $2 \times t_{PGCLK}$

...

1111:  $16 \times t_{PGCLK}$

*Note: These bits must not be modified when an acquisition is on-going.*

Bits 23:17 **SSD[6:0]**: Spread spectrum deviation

These bits are set and cleared by software. They define the spread spectrum deviation which consists in adding a variable number of periods of the SSCLK clock to the charge transfer pulse high state.

0000000:  $1 \times t_{SSCLK}$

0000001:  $2 \times t_{SSCLK}$

...

1111111:  $128 \times t_{SSCLK}$

*Note: These bits must not be modified when an acquisition is on-going.*

Bit 16 **SSE**: Spread spectrum enable

This bit is set and cleared by software to enable/disable the spread spectrum feature.

0: Spread spectrum disabled

1: Spread spectrum enabled

*Note: This bit must not be modified when an acquisition is on-going.*

Bit 15 **SSPSC**: Spread spectrum prescaler

This bit is set and cleared by software. It selects the AHB clock divider used to generate the spread spectrum clock (SSCLK).

0:  $f_{HCLK}$

1:  $f_{HCLK} / 2$

*Note: This bit must not be modified when an acquisition is on-going.*

Bits 14:12 **PGPSC[2:0]**: pulse generator prescaler

These bits are set and cleared by software. They select the AHB clock divider used to generate the pulse generator clock (PGCLK).

000:  $f_{HCLK}$

001:  $f_{HCLK} / 2$

010:  $f_{HCLK} / 4$

011:  $f_{HCLK} / 8$

100:  $f_{HCLK} / 16$

101:  $f_{HCLK} / 32$

110:  $f_{HCLK} / 64$

111:  $f_{HCLK} / 128$

*Note: These bits must not be modified when an acquisition is on-going.*

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **MCV[2:0]**: Max count value

These bits are set and cleared by software. They define the maximum number of charge transfer pulses that can be generated before a max count error is generated.

000: 255

001: 511

010: 1023

011: 2047

100: 4095

101: 8191

110: 16383

111: reserved

*Note: These bits must not be modified when an acquisition is on-going.*

Bit 4 **IODEF**: I/O Default mode

This bit is set and cleared by software. It defines the configuration of all the TSC I/Os when there is no on-going acquisition. When there is an on-going acquisition, it defines the configuration of all unused I/Os (not defined as sampling capacitor I/O or as channel I/O).

0: I/Os are forced to output push-pull low

1: I/Os are in input floating

*Note: This bit must not be modified when an acquisition is on-going.*

Bit 3 **SYNCPOL**: Synchronization pin polarity

This bit is set and cleared by software to select the polarity of the synchronization input pin.

0: Falling edge only

1: Rising edge and high level

Bit 2 **AM**: Acquisition mode

This bit is set and cleared by software to select the acquisition mode.

0: Normal acquisition mode (acquisition starts as soon as START bit is set)

1: Synchronized acquisition mode (acquisition starts if START bit is set and when the selected signal is detected on the SYNC input pin)

*Note: This bit must not be modified when an acquisition is on-going.*

Bit 1 **START**: Start a new acquisition

This bit is set by software to start a new acquisition. It is cleared by hardware as soon as the acquisition is complete or by software to cancel the on-going acquisition.

0: Acquisition not started

1: Start a new acquisition

Bit 0 **TSC**: Touch sensing controller enable

This bit is set and cleared by software to enable/disable the touch sensing controller.

0: Touch sensing controller disabled

1: Touch sensing controller enabled

*Note: When the touch sensing controller is disabled, TSC registers settings have no effect.*

### 27.6.2 TSC interrupt enable register (TSC\_IER)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bit 1 **MCEIE**: Max count error interrupt enable

This bit is set and cleared by software to enable/disable the max count error interrupt.

- 0: Max count error interrupt disabled
- 1: Max count error interrupt enabled

Bit 0 **EOAIE**: End of acquisition interrupt enable

This bit is set and cleared by software to enable/disable the end of acquisition interrupt.

- 0: End of acquisition interrupt disabled
- 1: End of acquisition interrupt enabled

### 27.6.3 TSC interrupt clear register (TSC\_ICR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 1 **MCEIC**: Max count error interrupt clear

This bit is set by software to clear the max count error flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

- 0: No effect
- 1: Clears the corresponding MCEF of the TSC\_ISR register

Bit 0 **EOAIC**: End of acquisition interrupt clear

This bit is set by software to clear the end of acquisition flag and it is cleared by hardware when the flag is reset. Writing a '0' has no effect.

- 0: No effect
- 1: Clears the corresponding EOAF of the TSC\_ISR register

### 27.6.4 TSC interrupt status register (TSC\_ISR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 1 **MCEF**: Max count error flag

This bit is set by hardware as soon as an analog I/O group counter reaches the max count value specified. It is cleared by software writing 1 to the bit MCEIC of the TSC\_ICR register.

0: No max count error (MCE) detected

1: Max count error (MCE) detected

Bit 0 **EOAF**: End of acquisition flag

This bit is set by hardware when the acquisition of all enabled group is complete (all GxS bits of all enabled analog I/O groups are set or when a max count error is detected). It is cleared by software writing 1 to the bit EOAF of the TSC\_ICR register.

0: Acquisition is on-going or not started

1: Acquisition is complete

### 27.6.5 TSC I/O hysteresis control register (TSC\_IOHCR)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
rw															

Bits 31:0 **Gx\_IOy**: Gx\_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx\_IOy Schmitt trigger hysteresis.

0: Gx\_IOy Schmitt trigger hysteresis disabled

1: Gx\_IOy Schmitt trigger hysteresis enabled

*Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).*

### 27.6.6 TSC I/O analog switch control register (TSC\_IOASCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
r/w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
r/w															

Bits 31:0 **Gx\_IOy**: Gx\_IOy analog switch enable

These bits are set and cleared by software to enable/disable the Gx\_IOy analog switch.

0: Gx\_IOy analog switch disabled (opened)

1: Gx\_IOy analog switch enabled (closed)

*Note: These bits control the I/O analog switch whatever the I/O control mode is (even if controlled by standard GPIO registers).*

### 27.6.7 TSC I/O sampling control register (TSC\_IOSCR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
r/w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
r/w															

Bits 31:0 **Gx\_IOy**: Gx\_IOy sampling mode

These bits are set and cleared by software to configure the Gx\_IOy as a sampling capacitor I/O. Only one I/O per analog I/O group must be defined as sampling capacitor.

0: Gx\_IOy unused

1: Gx\_IOy used as sampling capacitor

*Note: These bits must not be modified when an acquisition is on-going.*

*During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOSCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*

### 27.6.8 TSC I/O channel control register (TSC\_IOCCR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
r/w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
r/w															

Bits 31:0 **Gx\_IOy**: Gx\_IOy channel mode

These bits are set and cleared by software to configure the Gx\_IOy as a channel I/O.

0: Gx\_IOy unused

1: Gx\_IOy used as channel

*Note: These bits must not be modified when an acquisition is on-going.*

*During the acquisition phase and even if the TSC peripheral alternate function is not enabled, as soon as the TSC\_IOCCR bit is set, the corresponding GPIO analog switch is automatically controlled by the touch sensing controller.*

### 27.6.9 TSC I/O group control status register (TSC\_I OGCSR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	G8E	G7E	G6E	G5E	G4E	G3E	G2E	G1E							
								rw							

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

Bits 23:16 **GxS**: Analog I/O group x status

These bits are set by hardware when the acquisition on the corresponding enabled analog I/O group x is complete. They are cleared by hardware when a new acquisition is started.

0: Acquisition on analog I/O group x is on-going or not started

1: Acquisition on analog I/O group x is complete

*Note: When a max count error is detected the remaining GxS bits of the enabled analog I/O groups are not set.*

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

Bits 7:0 **GxE**: Analog I/O group x enable

These bits are set and cleared by software to enable/disable the acquisition (counter is counting) on the corresponding analog I/O group x.

0: Acquisition on analog I/O group x disabled

1: Acquisition on analog I/O group x enabled

### 27.6.10 TSC I/O group x counter register (TSC\_I OGxCR) (x = 1..8)

Address offset: 0x30 + 0x04 x Analog I/O group number

Reset value: 0x0000 0000

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

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

Bits 13:0 **CNT[13:0]**: Counter value

These bits represent the number of charge transfer cycles generated on the analog I/O group x to complete its acquisition (voltage across C<sub>S</sub> has reached the threshold).

27.6.11 TSC register map

Table 102. TSC 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							
0x0000	TSC_CR	CTPH[3:0]				CTPL[3:0]				SSD[6:0]								SSE	SSPSC		PGPSC[2:0]				Res.		Res.		Res.		Res.		MVCV [2:0]		IODEF		SYNCPOL	AM	START	TSCE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x0004	TSC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
	Reset value																																							
0x0008	TSC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
	Reset value																																							
0x000C	TSC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
	Reset value																																							
0x0010	TSC_IOHCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						
0x0014	Reserved																																							
0x0018	TSC_IOASCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x001C	Reserved																																							
0x0020	TSC_IOSCR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x0024	Reserved																																							
0x0028	TSC_IOCRR	G8_IO4	G8_IO3	G8_IO2	G8_IO1	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x002C	Reserved																																							
0x0030	TSC_IQGCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G8S	G7S	G6S	G5S	G4S	G3S	G2S	G1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x0034	TSC_IQG1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
	Reset value																																							
		CNT[13:0]																																						
		0																																						



Table 102. TSC 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	
0x0038	TSC_I0G2CR	Res.	CNT[13:0]																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	TSC_I0G3CR	Res.	CNT[13:0]																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0040	TSC_I0G4CR	Res.	CNT[13:0]																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	TSC_I0G5CR	Res.	CNT[13:0]																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	TSC_I0G6CR	Res.	CNT[13:0]																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	TSC_I0G7CR	Res.	CNT[13:0]																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	TSC_I0G8CR	Res.	CNT[13:0]																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 28 Controller area network (bxCAN)

### 28.1 Introduction

The **Basic Extended CAN** peripheral, named **bxCAN**, interfaces the CAN network. It supports the CAN protocols version 2.0A and B. It has been designed to manage a high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages.

For safety-critical applications, the CAN controller provides all hardware functions for supporting the CAN Time Triggered Communication option.

### 28.2 bxCAN main features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1 Mbit/s
- Supports the Time Triggered Communication option

#### Transmission

- Three transmit mailboxes
- Configurable transmit priority
- Time Stamp on SOF transmission

#### Reception

- Two receive FIFOs with three stages
- Scalable filter banks:
  - 14 filter banks
- Identifier list feature
- Configurable FIFO overrun
- Time Stamp on SOF reception

#### Time-triggered communication option

- Disable automatic retransmission mode
- 16-bit free running timer
- Time Stamp sent in last two data bytes

#### Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

## 28.3 bxCAN general description

In today's CAN applications, the number of nodes in a network is increasing and often several networks are linked together via gateways. Typically the number of messages in the system (and thus to be handled by each node) has significantly increased. In addition to the application messages, Network Management and Diagnostic messages have been introduced.

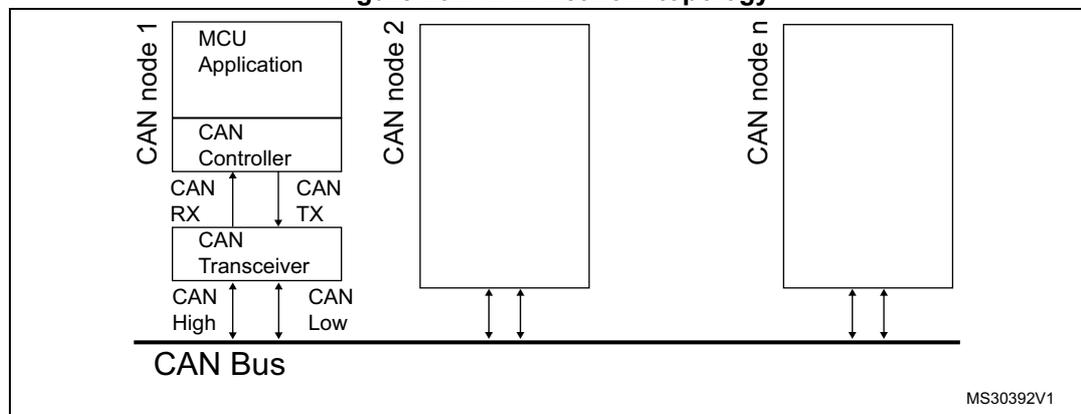
- An enhanced filtering mechanism is required to handle each type of message.

Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.

- A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.

The standard HLP (Higher Layer Protocol) based on standard CAN drivers requires an efficient interface to the CAN controller.

**Figure 262. CAN network topology**



### 28.3.1 CAN 2.0B active core

The bxCAN module handles the transmission and the reception of CAN messages fully autonomously. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported by hardware.

### 28.3.2 Control, status and configuration registers

The application uses these registers to:

- Configure CAN parameters, e.g. baud rate
- Request transmissions
- Handle receptions
- Manage interrupts
- Get diagnostic information

### 28.3.3 Tx mailboxes

Three transmit mailboxes are provided to the software for setting up messages. The transmission Scheduler decides which mailbox has to be transmitted first.

### 28.3.4 Acceptance filters

The bxCAN provides scalable/configurable identifier filter banks for selecting the incoming messages the software needs and discarding the others.

#### Receive FIFO

Two receive FIFOs are used by hardware to store the incoming messages. Three complete messages can be stored in each FIFO. The FIFOs are managed completely by hardware.

## 28.4 bxCAN operating modes

bxCAN has three main operating modes: **initialization**, **normal** and **Sleep**. After a hardware reset, bxCAN is in Sleep mode to reduce power consumption and an internal pull-up is active on CANTX. The software requests bxCAN to enter **initialization** or **Sleep** mode by setting the INRQ or SLEEP bits in the CAN\_MCR register. Once the mode has been entered, bxCAN confirms it by setting the INAK or SLAK bits in the CAN\_MSR register and the internal pull-up is disabled. When neither INAK nor SLAK are set, bxCAN is in **normal** mode. Before entering **normal** mode bxCAN always has to **synchronize** on the CAN bus. To synchronize, bxCAN waits until the CAN bus is idle, this means 11 consecutive recessive bits have been monitored on CANRX.

### 28.4.1 Initialization mode

The software initialization can be done while the hardware is in Initialization mode. To enter this mode the software sets the INRQ bit in the CAN\_MCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CAN\_MSR register.

To leave Initialization mode, the software clears the INQR bit. bxCAN has left Initialization mode once the INAK bit has been cleared by hardware.

While in Initialization Mode, all message transfers to and from the CAN bus are stopped and the status of the CAN bus output CANTX is recessive (high).

Entering Initialization Mode does not change any of the configuration registers.

To initialize the CAN Controller, software has to set up the Bit Timing (CAN\_BTR) and CAN options (CAN\_MCR) registers.

To initialize the registers associated with the CAN filter banks (mode, scale, FIFO assignment, activation and filter values), software has to set the FINIT bit (CAN\_FMR). Filter initialization also can be done outside the initialization mode.

*Note:* When FINIT=1, CAN reception is deactivated.

*The filter values also can be modified by deactivating the associated filter activation bits (in the CAN\_FA1R register).*

*If a filter bank is not used, it is recommended to leave it non active (leave the corresponding FACT bit cleared).*

### 28.4.2 Normal mode

Once the initialization is complete, the software must request the hardware to enter Normal mode to be able to synchronize on the CAN bus and start reception and transmission.

The request to enter Normal mode is issued by clearing the INRQ bit in the CAN\_MCR register. The bxCAN enters Normal mode and is ready to take part in bus activities when it has synchronized with the data transfer on the CAN bus. This is done by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle state). The switch to Normal mode is confirmed by the hardware by clearing the INAK bit in the CAN\_MSR register.

The initialization of the filter values is independent from Initialization Mode but must be done while the filter is not active (corresponding FACTx bit cleared). The filter scale and mode configuration must be configured before entering Normal Mode.

### 28.4.3 Sleep mode (low-power)

To reduce power consumption, bxCAN has a low-power mode called Sleep mode. This mode is entered on software request by setting the SLEEP bit in the CAN\_MCR register. In this mode, the bxCAN clock is stopped, however software can still access the bxCAN mailboxes.

If software requests entry to **initialization** mode by setting the INRQ bit while bxCAN is in **Sleep** mode, it must also clear the SLEEP bit.

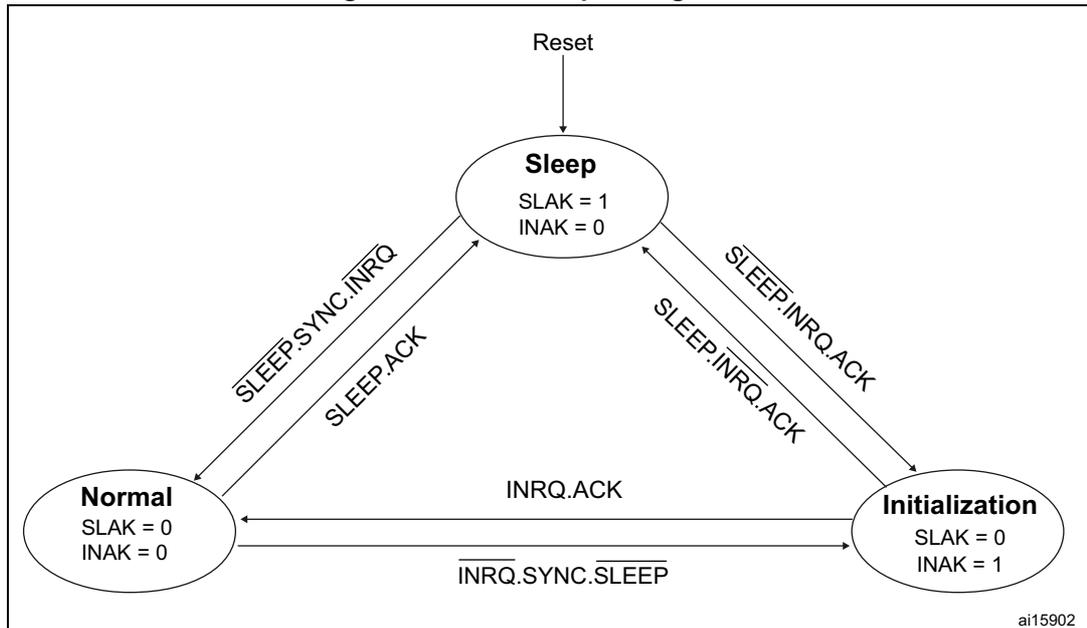
bxCAN can be woken up (exit Sleep mode) either by software clearing the SLEEP bit or on detection of CAN bus activity.

On CAN bus activity detection, hardware automatically performs the wakeup sequence by clearing the SLEEP bit if the AWUM bit in the CAN\_MCR register is set. If the AWUM bit is cleared, software has to clear the SLEEP bit when a wakeup interrupt occurs, in order to exit from Sleep mode.

*Note: If the wakeup interrupt is enabled (WKUIE bit set in CAN\_IER register) a wakeup interrupt will be generated on detection of CAN bus activity, even if the bxCAN automatically performs the wakeup sequence.*

After the SLEEP bit has been cleared, Sleep mode is exited once bxCAN has synchronized with the CAN bus, refer to [Figure 263: bxCAN operating modes](#). The Sleep mode is exited once the SLAK bit has been cleared by hardware.

Figure 263. bxCAN operating modes



1. ACK = The wait state during which hardware confirms a request by setting the INAK or SLAK bits in the CAN\_MSR register
2. SYNC = The state during which bxCAN waits until the CAN bus is idle, meaning 11 consecutive recessive bits have been monitored on CANRX

## 28.5 Test mode

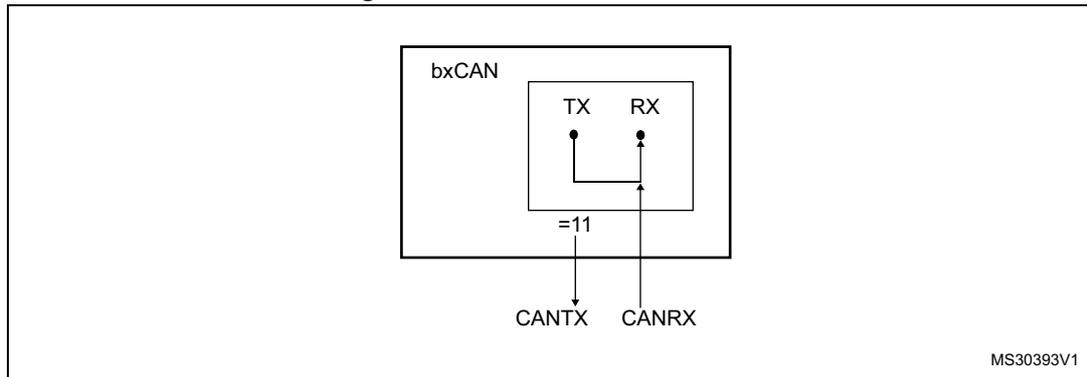
Test mode can be selected by the SILM and LBKM bits in the CAN\_BTR register. These bits must be configured while bxCAN is in Initialization mode. Once test mode has been selected, the INRQ bit in the CAN\_MCR register must be reset to enter Normal mode.

### 28.5.1 Silent mode

The bxCAN can be put in Silent mode by setting the SILM bit in the CAN\_BTR register.

In Silent mode, the bxCAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the bxCAN has to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. Silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames).

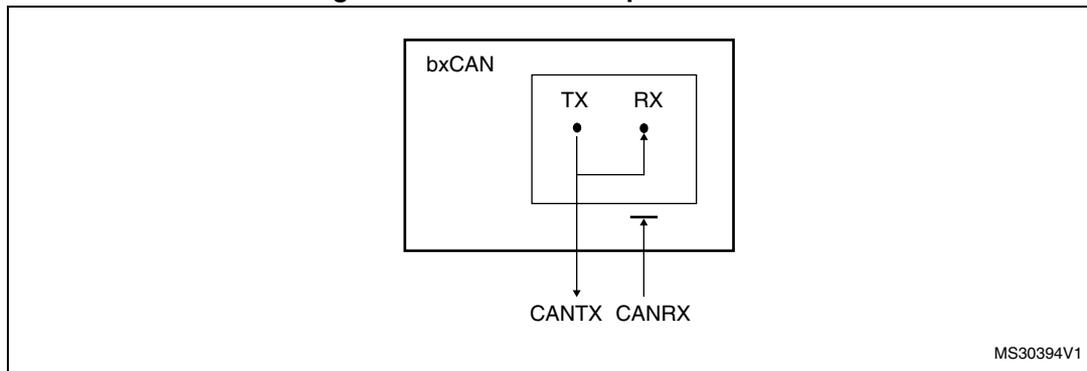
Figure 264. bxCAN in silent mode



28.5.2 Loop back mode

The bxCAN can be set in Loop Back Mode by setting the LBKM bit in the CAN\_BTR register. In Loop Back Mode, the bxCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) in a Receive mailbox.

Figure 265. bxCAN in loop back mode

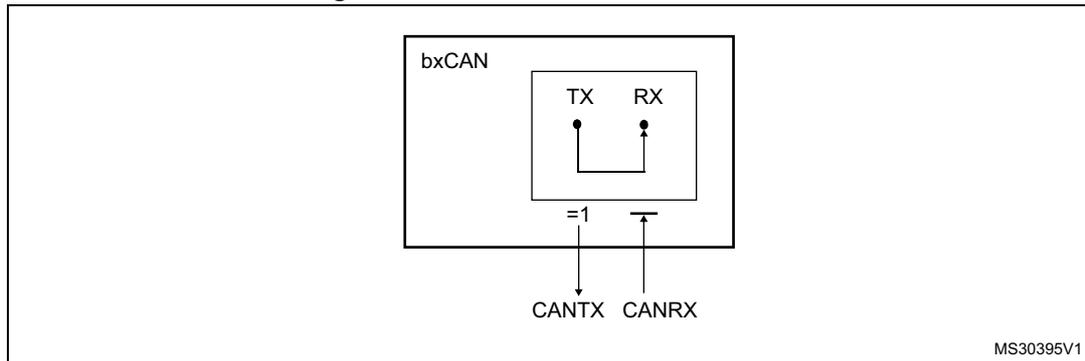


This mode is provided for self-test functions. To be independent of external events, the CAN Core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data / remote frame) in Loop Back Mode. In this mode, the bxCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the bxCAN. The transmitted messages can be monitored on the CANTX pin.

28.5.3 Loop back combined with silent mode

It is also possible to combine Loop Back mode and Silent mode by setting the LBKM and SILM bits in the CAN\_BTR register. This mode can be used for a “Hot Selftest”, meaning the bxCAN can be tested like in Loop Back mode but without affecting a running CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the bxCAN and the CANTX pin is held recessive.

Figure 266. bxCAN in combined mode



## 28.6 Behavior in Debug mode

When the microcontroller enters the debug mode (Cortex<sup>®</sup>-M4 with FPU core halted), the bxCAN continues to work normally or stops, depending on:

- the DBG\_CAN\_STOP bit in the DBG module.
- the DBF bit in CAN\_MCR. For more details, refer to [Section 28.9.2: CAN control and status registers](#).

## 28.7 bxCAN functional description

### 28.7.1 Transmission handling

In order to transmit a message, the application must select one **empty** transmit mailbox, set up the identifier, the data length code (DLC) and the data before requesting the transmission by setting the corresponding TXRQ bit in the CAN\_TlRxR register. Once the mailbox has left **empty** state, the software no longer has write access to the mailbox registers. Immediately after the TXRQ bit has been set, the mailbox enters **pending** state and waits to become the highest priority mailbox, see *Transmit Priority*. As soon as the mailbox has the highest priority it will be **scheduled** for transmission. The transmission of the message of the scheduled mailbox will start (enter **transmit** state) when the CAN bus becomes idle. Once the mailbox has been successfully transmitted, it will become **empty** again. The hardware indicates a successful transmission by setting the RQCP and TXOK bits in the CAN\_TSR register.

If the transmission fails, the cause is indicated by the ALST bit in the CAN\_TSR register in case of an Arbitration Lost, and/or the TERR bit, in case of transmission error detection.

#### Transmit priority

By identifier

When more than one transmit mailbox is pending, the transmission order is given by the identifier of the message stored in the mailbox. The message with the lowest identifier value has the highest priority according to the arbitration of the CAN protocol. If the identifier values are equal, the lower mailbox number will be scheduled first.

By transmit request order

The transmit mailboxes can be configured as a transmit FIFO by setting the TXFP bit in the CAN\_MCR register. In this mode the priority order is given by the transmit request order.

This mode is very useful for segmented transmission.

### Abort

A transmission request can be aborted by the user setting the ABRQ bit in the CAN\_TSR register. In **pending** or **scheduled** state, the mailbox is aborted immediately. An abort request while the mailbox is in **transmit** state can have two results. If the mailbox is transmitted successfully the mailbox becomes **empty** with the TXOK bit set in the CAN\_TSR register. If the transmission fails, the mailbox becomes **scheduled**, the transmission is aborted and becomes **empty** with TXOK cleared. In all cases the mailbox will become **empty** again at least at the end of the current transmission.

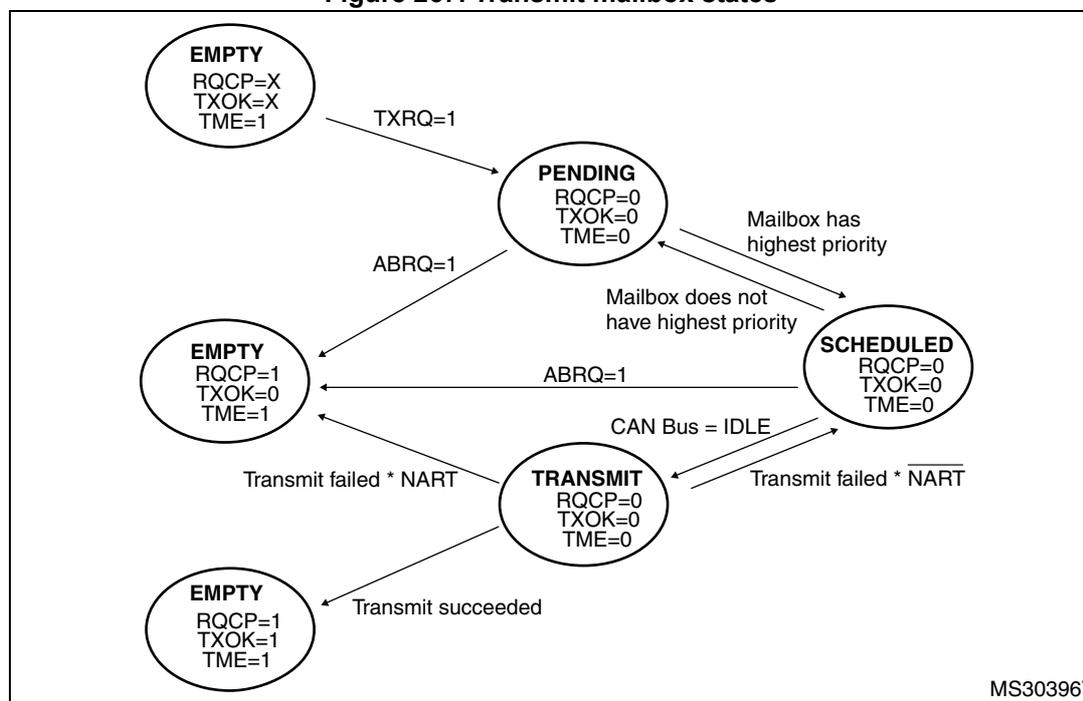
### Nonautomatic retransmission mode

This mode has been implemented in order to fulfil the requirement of the Time Triggered Communication option of the CAN standard. To configure the hardware in this mode the NART bit in the CAN\_MCR register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the message transmission.

At the end of the first transmission attempt, the hardware considers the request as completed and sets the RQCP bit in the CAN\_TSR register. The result of the transmission is indicated in the CAN\_TSR register by the TXOK, ALST and TERR bits.

Figure 267. Transmit mailbox states



MS30396V

### 28.7.2 Time triggered communication mode

In this mode, the internal counter of the CAN hardware is activated and used to generate the Time Stamp value stored in the CAN\_RDTxR/CAN\_TDTxR registers, respectively (for Rx and Tx mailboxes). The internal counter is incremented each CAN bit time (refer to [Section 28.7.7: Bit timing](#)). The internal counter is captured on the sample point of the Start Of Frame bit in both reception and transmission.

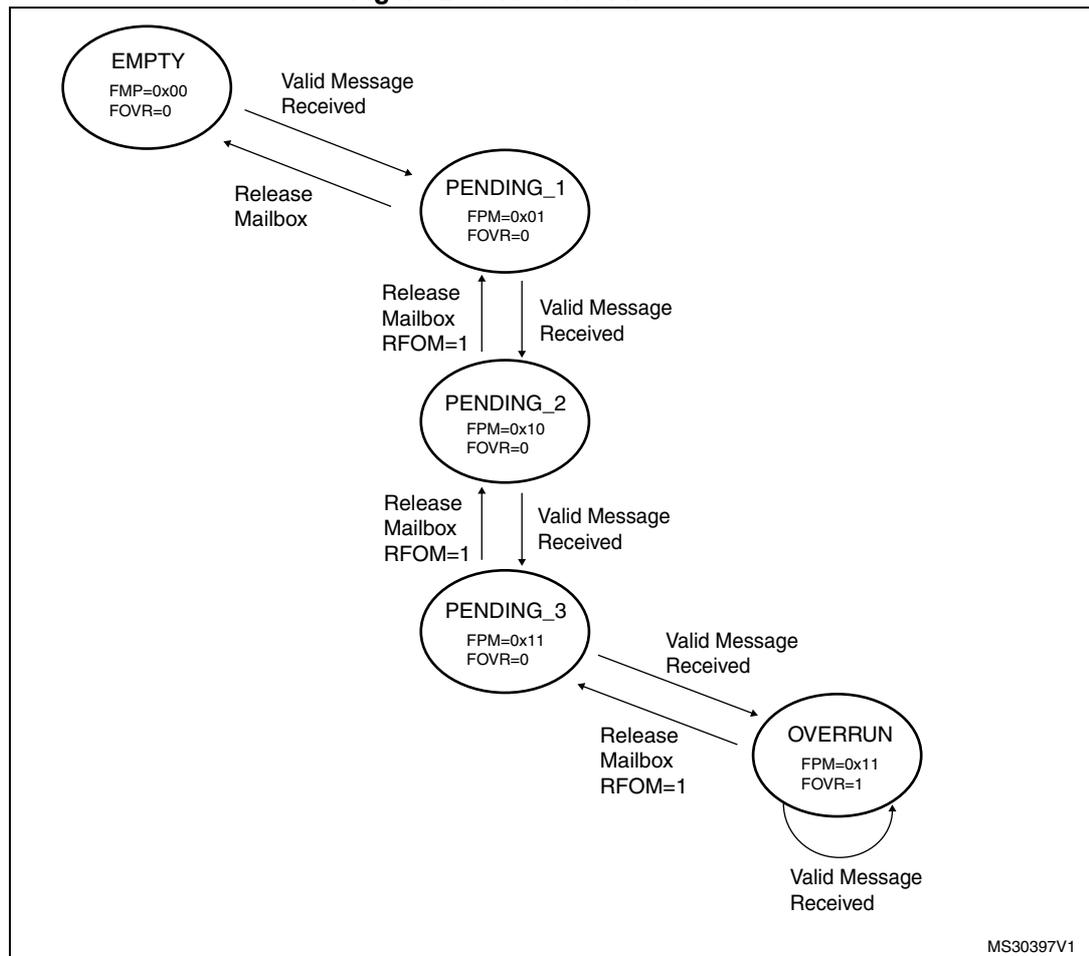
### 28.7.3 Reception handling

For the reception of CAN messages, three mailboxes organized as a FIFO are provided. In order to save CPU load, simplify the software and guarantee data consistency, the FIFO is managed completely by hardware. The application accesses the messages stored in the FIFO through the FIFO output mailbox.

#### Valid message

A received message is considered as valid **when** it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) **and** It passed through the identifier filtering successfully, see [Section 28.7.4: Identifier filtering](#).

Figure 268. Receive FIFO states



## FIFO management

Starting from the **empty** state, the first valid message received is stored in the FIFO which becomes **pending\_1**. The hardware signals the event setting the FMP[1:0] bits in the CAN\_RFR register to the value 01b. The message is available in the FIFO output mailbox. The software reads out the mailbox content and releases it by setting the RFOM bit in the CAN\_RFR register. The FIFO becomes **empty** again. If a new valid message has been received in the meantime, the FIFO stays in **pending\_1** state and the new message is available in the output mailbox.

If the application does not release the mailbox, the next valid message will be stored in the FIFO which enters **pending\_2** state (FMP[1:0] = 10b). The storage process is repeated for the next valid message putting the FIFO into **pending\_3** state (FMP[1:0] = 11b). At this point, the software must release the output mailbox by setting the RFOM bit, so that a mailbox is free to store the next valid message. Otherwise the next valid message received will cause a loss of message.

Refer also to [Section 28.7.5: Message storage](#)

## Overrun

Once the FIFO is in **pending\_3** state (i.e. the three mailboxes are full) the next valid message reception will lead to an **overrun** and a message will be lost. The hardware signals the overrun condition by setting the FOVR bit in the CAN\_RFR register. Which message is lost depends on the configuration of the FIFO:

- If the FIFO lock function is disabled (RFLM bit in the CAN\_MCR register cleared) the last message stored in the FIFO will be overwritten by the new incoming message. In this case the latest messages will be always available to the application.
- If the FIFO lock function is enabled (RFLM bit in the CAN\_MCR register set) the most recent message will be discarded and the software will have the three oldest messages in the FIFO available.

## Reception related interrupts

Once a message has been stored in the FIFO, the FMP[1:0] bits are updated and an interrupt request is generated if the FMPIE bit in the CAN\_IER register is set.

When the FIFO becomes full (i.e. a third message is stored) the FULL bit in the CAN\_RFR register is set and an interrupt is generated if the FFIE bit in the CAN\_IER register is set.

On overrun condition, the FOVR bit is set and an interrupt is generated if the FOVIE bit in the CAN\_IER register is set.

### 28.7.4 Identifier filtering

In the CAN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On message reception a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the SRAM. If not, the message must be discarded without intervention by the software.

To fulfill this requirement, the bxCAN Controller provides 28 configurable and scalable filter banks (27-0) to the application. In other devices the bxCAN Controller provides 14

configurable and scalable filter banks (13-0) to the application in order to receive only the messages the software needs. This hardware filtering saves CPU resources which would be otherwise needed to perform filtering by software. Each filter bank x consists of two 32-bit registers, CAN\_FxR0 and CAN\_FxR1.

### Scalable width

To optimize and adapt the filters to the application needs, each filter bank can be scaled independently. Depending on the filter scale a filter bank provides:

- One 32-bit filter for the STDID[10:0], EXTID[17:0], IDE and RTR bits.
- Two 16-bit filters for the STDID[10:0], RTR, IDE and EXTID[17:15] bits.

Refer to [Figure 269](#).

Furthermore, the filters can be configured in mask mode or in identifier list mode.

### Mask mode

In **mask** mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”.

### Identifier list mode

In **identifier list** mode, the mask registers are used as identifier registers. Thus instead of defining an identifier and a mask, two identifiers are specified, doubling the number of single identifiers. All bits of the incoming identifier must match the bits specified in the filter registers.

### Filter bank scale and mode configuration

The filter banks are configured by means of the corresponding CAN\_FMR register. To configure a filter bank it must be deactivated by clearing the FACT bit in the CAN\_FAR register. The filter scale is configured by means of the corresponding FSCx bit in the CAN\_FS1R register, refer to [Figure 269](#). The **identifier list** or **identifier mask** mode for the corresponding Mask/Identifier registers is configured by means of the FBMx bits in the CAN\_FMR register.

To filter a group of identifiers, configure the Mask/Identifier registers in mask mode.

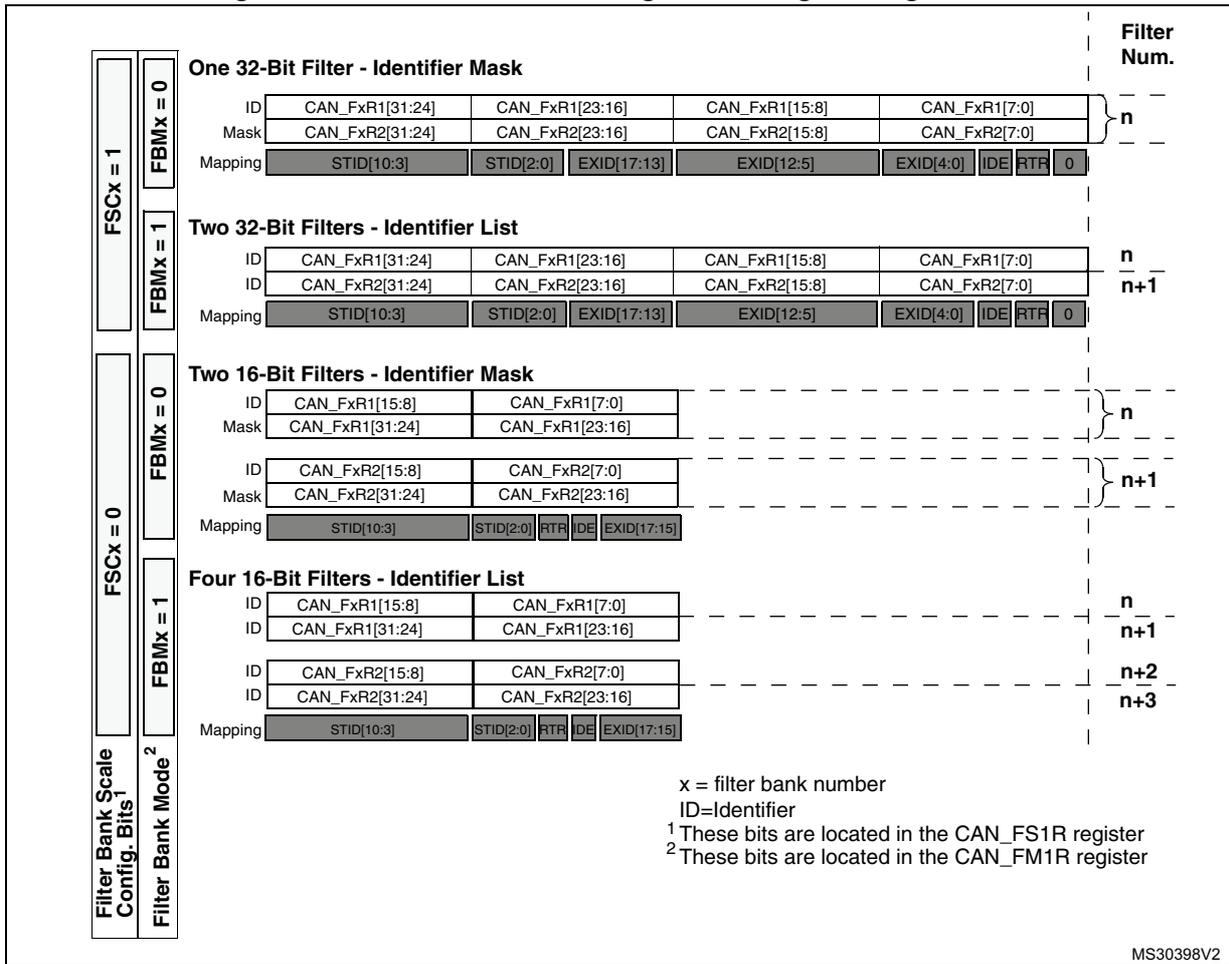
To select single identifiers, configure the Mask/Identifier registers in identifier list mode.

Filters not used by the application should be left deactivated.

Each filter within a filter bank is numbered (called the *Filter Number*) from 0 to a maximum dependent on the mode and the scale of each of the filter banks.

Concerning the filter configuration, refer to [Figure 269](#).

Figure 269. Filter bank scale configuration - register organization



**Filter match index**

Once a message has been received in the FIFO it is available to the application. Typically, application data is copied into SRAM locations. To copy the data to the right location the application has to identify the data by means of the identifier. To avoid this, and to ease the access to the SRAM locations, the CAN controller provides a Filter Match Index.

This index is stored in the mailbox together with the message according to the filter priority rules. Thus each received message has its associated filter match index.

The Filter Match index can be used in two ways:

- Compare the Filter Match index with a list of expected values.
- Use the Filter Match Index as an index on an array to access the data destination location.

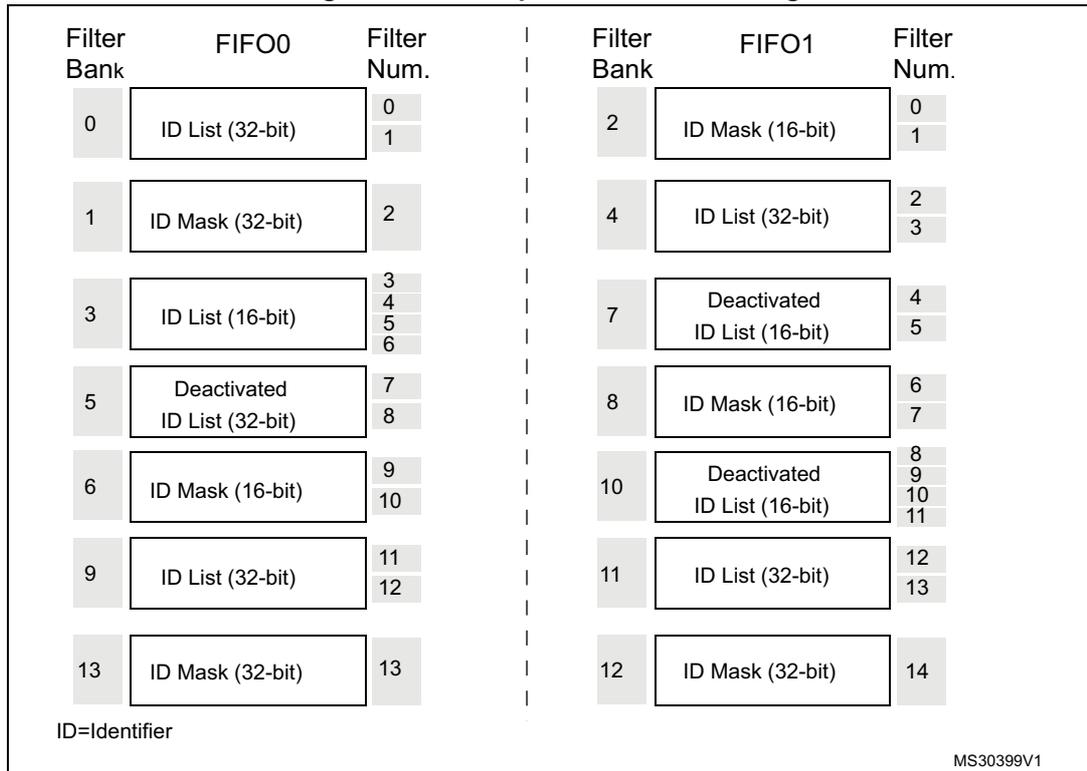
For nonmasked filters, the software no longer has to compare the identifier.

If the filter is masked the software reduces the comparison to the masked bits only.

The index value of the filter number does not take into account the activation state of the filter banks. In addition, two independent numbering schemes are used, one for each FIFO. Refer to *Figure 270* for an example.



Figure 270. Example of filter numbering

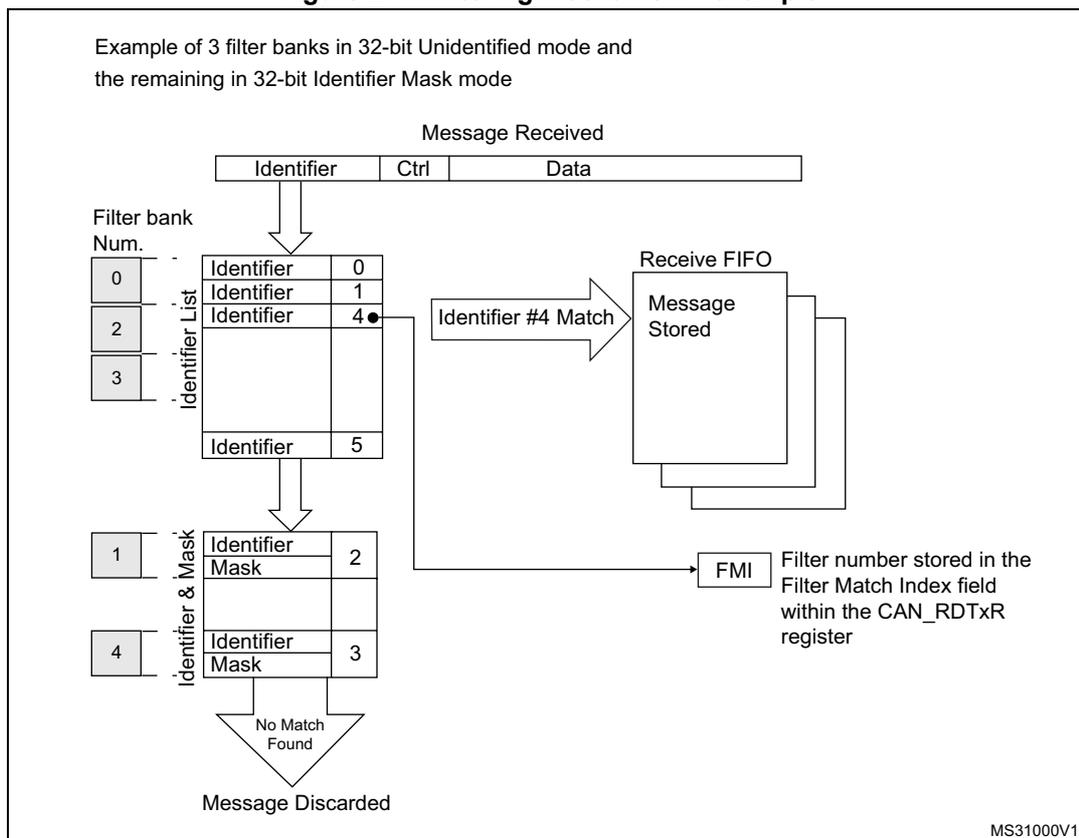


**Filter priority rules**

Depending on the filter combination it may occur that an identifier passes successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following priority rules:

- A 32-bit filter takes priority over a 16-bit filter.
- For filters of equal scale, priority is given to the Identifier List mode over the Identifier Mask mode
- For filters of equal scale and mode, priority is given by the filter number (the lower the number, the higher the priority).

Figure 271. Filtering mechanism - example



The example above shows the filtering principle of the bxCAN. On reception of a message, the identifier is compared first with the filters configured in identifier list mode. If there is a match, the message is stored in the associated FIFO and the index of the matching filter is stored in the Filter Match Index. As shown in the example, the identifier matches with Identifier #2 thus the message content and FMI 2 is stored in the FIFO.

If there is no match, the incoming identifier is then compared with the filters configured in mask mode.

If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without disturbing the software.

### 28.7.5 Message storage

The interface between the software and the hardware for the CAN messages is implemented by means of mailboxes. A mailbox contains all information related to a message; identifier, data, control, status and time stamp information.

#### Transmit mailbox

The software sets up the message to be transmitted in an empty transmit mailbox. The status of the transmission is indicated by hardware in the CAN\_TSR register.

**Table 103. Transmit mailbox mapping**

Offset to transmit mailbox base address	Register name
0	CAN_TlRxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

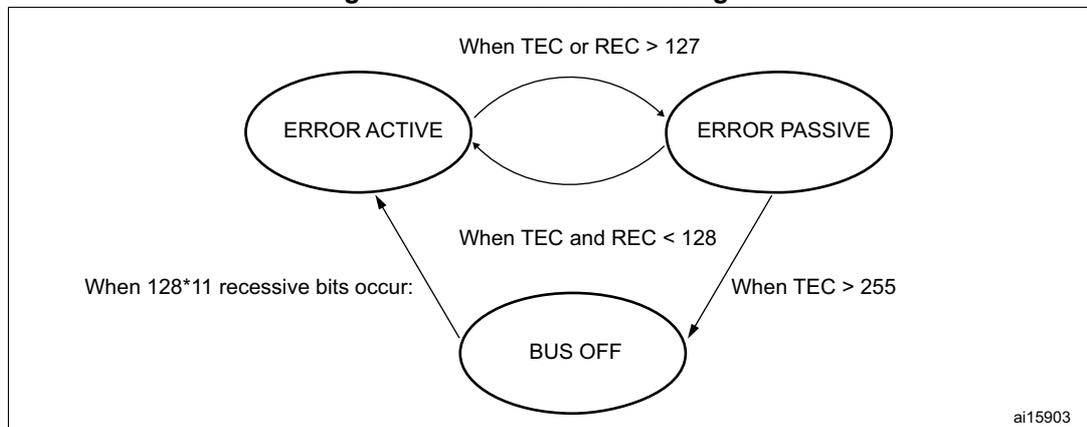
**Receive mailbox**

When a message has been received, it is available to the software in the FIFO output mailbox. Once the software has handled the message (e.g. read it) the software must release the FIFO output mailbox by means of the RFOM bit in the CAN\_RFR register to make the next incoming message available. The filter match index is stored in the MFMI field of the CAN\_RDTxR register. The 16-bit time stamp value is stored in the TIME[15:0] field of CAN\_RDTxR.

**Table 104. Receive mailbox mapping**

Offset to receive mailbox base address (bytes)	Register name
0	CAN_RlRxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

**Figure 272. CAN error state diagram**



## 28.7.6 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TEC value, in CAN\_ESR register) and a Receive Error Counter (REC value, in the CAN\_ESR register), which get incremented or decremented according to the error condition. For detailed information about TEC and REC management, please refer to the CAN standard.

Both of them may be read by software to determine the stability of the network. Furthermore, the CAN hardware provides detailed information on the current error status in CAN\_ESR register. By means of the CAN\_IER register (ERRIE bit, etc.), the software can configure the interrupt generation on error detection in a very flexible way.

### Bus-Off recovery

The Bus-Off state is reached when TEC is greater than 255, this state is indicated by BOFF bit in CAN\_ESR register. In Bus-Off state, the bxCAN is no longer able to transmit and receive messages.

Depending on the ABOM bit in the CAN\_MCR register bxCAN will recover from Bus-Off (become error active again) either automatically or on software request. But in both cases the bxCAN has to wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CANRX).

If ABOM is set, the bxCAN will start the recovering sequence automatically after it has entered Bus-Off state.

If ABOM is cleared, the software must initiate the recovering sequence by requesting bxCAN to enter and to leave initialization mode.

*Note:* In initialization mode, bxCAN does not monitor the CANRX signal, therefore it cannot complete the recovery sequence. **To recover, bxCAN must be in normal mode.**

## 28.7.7 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

- **Synchronization segment (SYNC\_SEG):** a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ( $1 \times t_{CAN}$ ).
- **Bit segment 1 (BS1):** defines the location of the sample point. It includes the PROP\_SEG and PHASE\_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- **Bit segment 2 (BS2):** defines the location of the transmit point. It represents the PHASE\_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC\_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC\_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit Timing Register (CAN\_BTR) is only possible while the device is in Standby mode.

*Note: For a detailed description of the CAN bit timing and resynchronization mechanism, please refer to the ISO 11898 standard.*

**Figure 273. Bit timing**

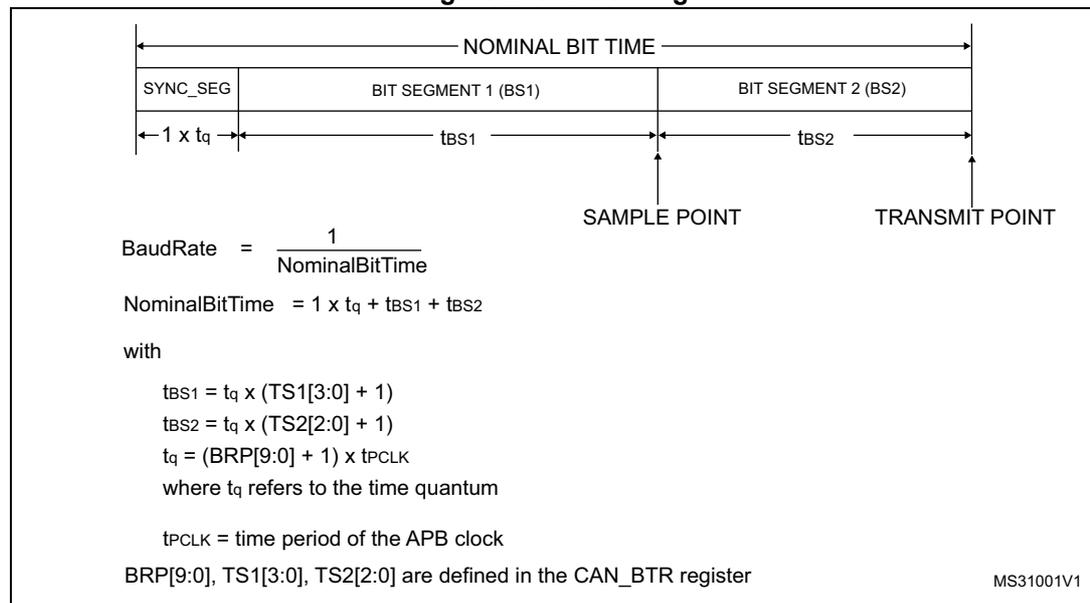
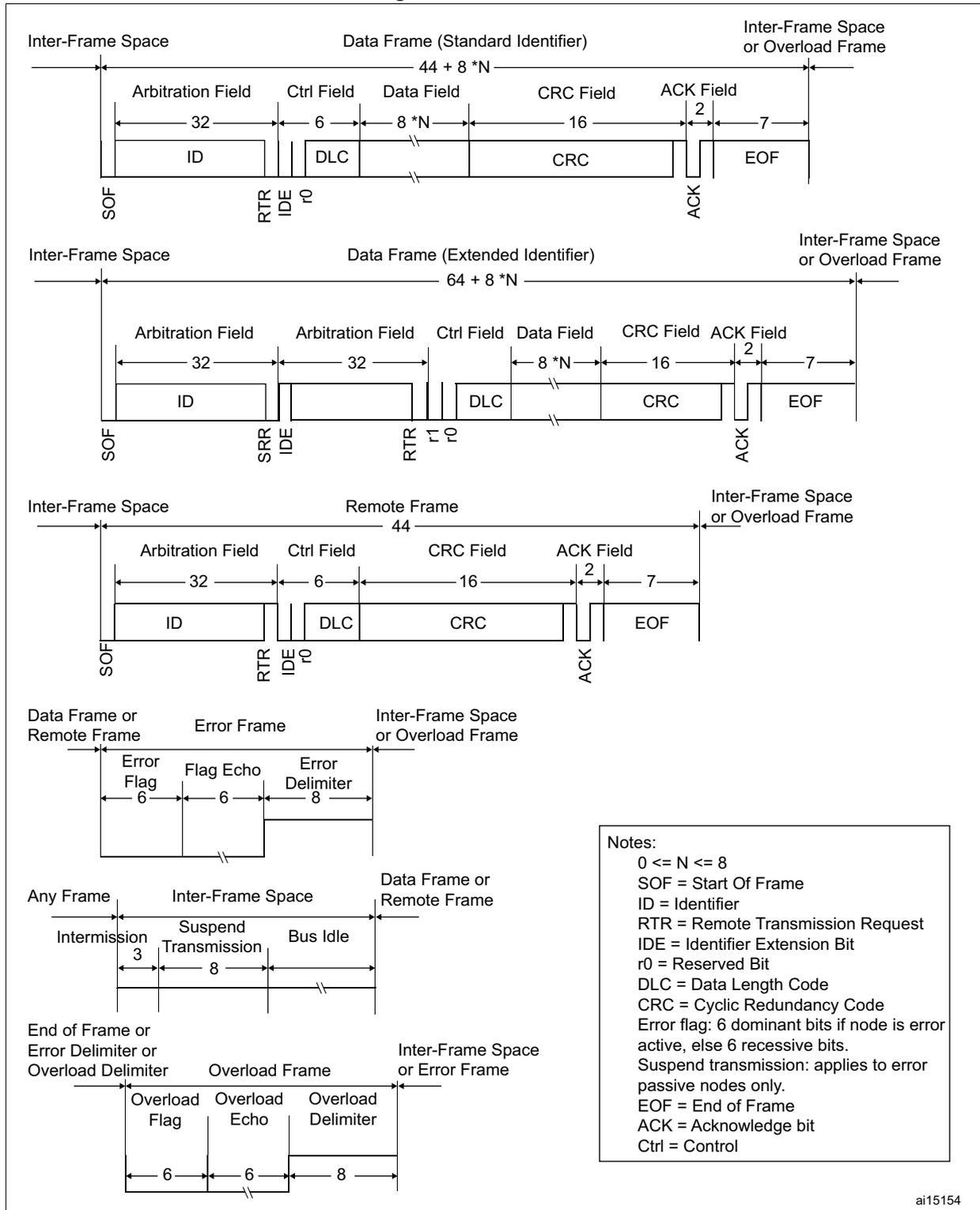


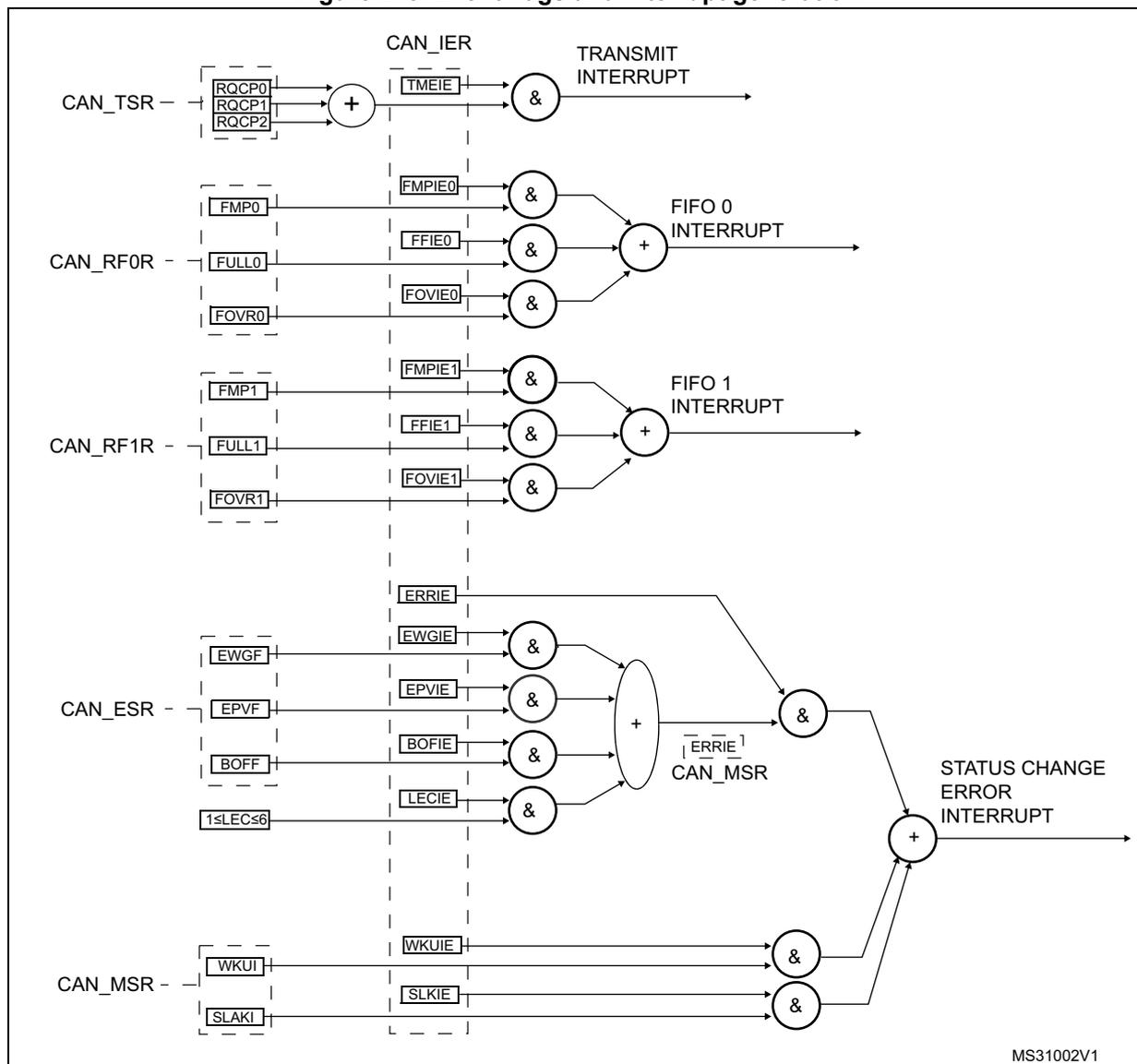
Figure 274. CAN frames



## 28.8 bxCAN interrupts

Four interrupt vectors are dedicated to bxCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CAN\_IER).

Figure 275. Event flags and interrupt generation



- The **transmit interrupt** can be generated by the following events:
  - Transmit mailbox 0 becomes empty, RQCP0 bit in the CAN\_TSR register set.
  - Transmit mailbox 1 becomes empty, RQCP1 bit in the CAN\_TSR register set.
  - Transmit mailbox 2 becomes empty, RQCP2 bit in the CAN\_TSR register set.
- The **FIFO 0 interrupt** can be generated by the following events:
  - Reception of a new message, FMP0 bits in the CAN\_RF0R register are not '00'.
  - FIFO0 full condition, FULL0 bit in the CAN\_RF0R register set.
  - FIFO0 overrun condition, FOVR0 bit in the CAN\_RF0R register set.

- The **FIFO 1 interrupt** can be generated by the following events:
  - Reception of a new message, FMP1 bits in the CAN\_RF1R register are not '00'.
  - FIFO1 full condition, FULL1 bit in the CAN\_RF1R register set.
  - FIFO1 overrun condition, FOVR1 bit in the CAN\_RF1R register set.
- The **error and status change interrupt** can be generated by the following events:
  - Error condition, for more details on error conditions please refer to the CAN Error Status register (CAN\_ESR).
  - Wakeup condition, SOF monitored on the CAN Rx signal.
  - Entry into Sleep mode.

## 28.9 CAN registers

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

### 28.9.1 Register access protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the CAN\_BTR register can be modified by software only while the CAN hardware is in initialization mode.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state, refer to [Figure 267: Transmit mailbox states](#).

The filter values can be modified either deactivating the associated filter banks or by setting the FINIT bit. Moreover, the modification of the filter configuration (scale, mode and FIFO assignment) in CAN\_FmRxR, CAN\_FSxR and CAN\_FFAR registers can only be done when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

### 28.9.2 CAN control and status registers

Refer to [Section 1.1](#) for a list of abbreviations used in register descriptions.

#### CAN master control register (CAN\_MCR)

Address offset: 0x00

Reset value: 0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBF
															rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Res	TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ						
rs								rW	rW						

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

Bit 16 **DBF**: Debug freeze

0: CAN working during debug

1: CAN reception/transmission frozen during debug. Reception FIFOs can still be accessed/controlled normally.

Bit 15 **RESET**: bxCAN software master reset

0: Normal operation.

1: Force a master reset of the bxCAN -> Sleep mode activated after reset (FMP bits and CAN\_MCR register are initialized to the reset values). This bit is automatically reset to 0.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **TTCM**: Time triggered communication mode

0: Time Triggered Communication mode disabled.

1: Time Triggered Communication mode enabled

*Note: For more information on Time Triggered Communication mode, please refer to [Section 28.7.2: Time triggered communication mode](#).*

Bit 6 **ABOM**: Automatic bus-off management

This bit controls the behavior of the CAN hardware on leaving the Bus-Off state.

0: The Bus-Off state is left on software request, once 128 occurrences of 11 recessive bits have been monitored and the software has first set and cleared the INRQ bit of the CAN\_MCR register.

1: The Bus-Off state is left automatically by hardware once 128 occurrences of 11 recessive bits have been monitored.

For detailed information on the Bus-Off state please refer to [Section 28.7.6: Error management](#).

Bit 5 **AWUM**: Automatic wakeup mode

This bit controls the behavior of the CAN hardware on message reception during Sleep mode.

0: The Sleep mode is left on software request by clearing the SLEEP bit of the CAN\_MCR register.

1: The Sleep mode is left automatically by hardware on CAN message detection.

The SLEEP bit of the CAN\_MCR register and the SLAK bit of the CAN\_MSR register are cleared by hardware.

Bit 4 **NART**: No automatic retransmission

0: The CAN hardware will automatically retransmit the message until it has been successfully transmitted according to the CAN standard.

1: A message will be transmitted only once, independently of the transmission result (successful, error or arbitration lost).

Bit 3 **RFLM**: Receive FIFO locked mode

0: Receive FIFO not locked on overrun. Once a receive FIFO is full the next incoming message will overwrite the previous one.

1: Receive FIFO locked against overrun. Once a receive FIFO is full the next incoming message will be discarded.

Bit 2 **TXFP**: Transmit FIFO priority

This bit controls the transmission order when several mailboxes are pending at the same time.

0: Priority driven by the identifier of the message

1: Priority driven by the request order (chronologically)

Bit 1 **SLEEP**: Sleep mode request

This bit is set by software to request the CAN hardware to enter the Sleep mode. Sleep mode will be entered as soon as the current CAN activity (transmission or reception of a CAN frame) has been completed.

This bit is cleared by software to exit Sleep mode.

This bit is cleared by hardware when the AWUM bit is set and a SOF bit is detected on the CAN Rx signal.

This bit is set after reset - CAN starts in Sleep mode.

Bit 0 **INRQ**: Initialization request

The software clears this bit to switch the hardware into normal mode. Once 11 consecutive recessive bits have been monitored on the Rx signal the CAN hardware is synchronized and ready for transmission and reception. Hardware signals this event by clearing the INAK bit in the CAN\_MSR register.

Software sets this bit to request the CAN hardware to enter initialization mode. Once software has set the INRQ bit, the CAN hardware waits until the current CAN activity (transmission or reception) is completed before entering the initialization mode. Hardware signals this event by setting the INAK bit in the CAN\_MSR register.

**CAN master status register (CAN\_MSR)**

Address offset: 0x04

Reset value: 0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	RX	SAMP	RXM	TXM	Res	Res	Res	SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RX**: CAN Rx signal

Monitors the actual value of the **CAN\_RX** Pin.

Bit 10 **SAMP**: Last sample point

The value of RX on the last sample point (current received bit value).

Bit 9 **RXM**: Receive mode

The CAN hardware is currently receiver.

Bit 8 **TXM**: Transmit mode

The CAN hardware is currently transmitter.

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

Bit 4 **SLAKI**: Sleep acknowledge interrupt

When SLKIE=1, this bit is set by hardware to signal that the bxCAN has entered Sleep Mode. When set, this bit generates a status change interrupt if the SLKIE bit in the CAN\_IER register is set.

This bit is cleared by software or by hardware, when SLAK is cleared.

*Note: When SLKIE=0, no polling on SLAKI is possible. In this case the SLAK bit can be polled.*

Bit 3 **WKUI**: Wakeup interrupt

This bit is set by hardware to signal that a SOF bit has been detected while the CAN hardware was in Sleep mode. Setting this bit generates a status change interrupt if the WKUIE bit in the CAN\_IER register is set.

This bit is cleared by software.

Bit 2 **ERRI**: Error interrupt

This bit is set by hardware when a bit of the CAN\_ESR has been set on error detection and the corresponding interrupt in the CAN\_IER is enabled. Setting this bit generates a status change interrupt if the ERRIE bit in the CAN\_IER register is set.

This bit is cleared by software.

Bit 1 **SLAK**: Sleep acknowledge

This bit is set by hardware and indicates to the software that the CAN hardware is now in Sleep mode. This bit acknowledges the Sleep mode request from the software (set SLEEP bit in CAN\_MCR register).

This bit is cleared by hardware when the CAN hardware has left Sleep mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

*Note: The process of leaving Sleep mode is triggered when the SLEEP bit in the CAN\_MCR register is cleared. Please refer to the AWUM bit of the CAN\_MCR register description for detailed information for clearing SLEEP bit*

Bit 0 **INAK**: Initialization acknowledge

This bit is set by hardware and indicates to the software that the CAN hardware is now in initialization mode. This bit acknowledges the initialization request from the software (set INRQ bit in CAN\_MCR register).

This bit is cleared by hardware when the CAN hardware has left the initialization mode (to be synchronized on the CAN bus). To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

**CAN transmit status register (CAN\_TSR)**

Address offset: 0x08

Reset value: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Res	Res	Res	TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Res	Res	Res	TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res	Res	Res	TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

- Bit 31 **LOW2**: Lowest priority flag for mailbox 2  
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 2 has the lowest priority.
- Bit 30 **LOW1**: Lowest priority flag for mailbox 1  
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 1 has the lowest priority.
- Bit 29 **LOW0**: Lowest priority flag for mailbox 0  
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 0 has the lowest priority.  
*Note: The LOW[2:0] bits are set to zero when only one mailbox is pending.*
- Bit 28 **TME2**: Transmit mailbox 2 empty  
This bit is set by hardware when no transmit request is pending for mailbox 2.
- Bit 27 **TME1**: Transmit mailbox 1 empty  
This bit is set by hardware when no transmit request is pending for mailbox 1.
- Bit 26 **TME0**: Transmit mailbox 0 empty  
This bit is set by hardware when no transmit request is pending for mailbox 0.
- Bits 25:24 **CODE[1:0]**: Mailbox code  
In case at least one transmit mailbox is free, the code value is equal to the number of the next transmit mailbox free.  
In case all transmit mailboxes are pending, the code value is equal to the number of the transmit mailbox with the lowest priority.
- Bit 23 **ABRQ2**: Abort request for mailbox 2  
Set by software to abort the transmission request for the corresponding mailbox.  
Cleared by hardware when the mailbox becomes empty.  
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 22:20 Reserved, must be kept at reset value.
- Bit 19 **TERR2**: Transmission error of mailbox 2  
This bit is set when the previous TX failed due to an error.
- Bit 18 **ALST2**: Arbitration lost for mailbox 2  
This bit is set when the previous TX failed due to an arbitration lost.
- Bit 17 **TXOK2**: Transmission OK of mailbox 2  
The hardware updates this bit after each transmission attempt.  
0: The previous transmission failed  
1: The previous transmission was successful  
This bit is set by hardware when the transmission request on mailbox 2 has been completed successfully. Please refer to [Figure 267](#).
- Bit 16 **RQCP2**: Request completed mailbox2  
Set by hardware when the last request (transmit or abort) has been performed.  
Cleared by software writing a "1" or by hardware on transmission request (TXRQ2 set in CAN\_TMD2R register).  
Clearing this bit clears all the status bits (TXOK2, ALST2 and TERR2) for Mailbox 2.
- Bit 15 **ABRQ1**: Abort request for mailbox 1  
Set by software to abort the transmission request for the corresponding mailbox.  
Cleared by hardware when the mailbox becomes empty.  
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 14:12 Reserved, must be kept at reset value.

- Bit 11 **TERR1**: Transmission error of mailbox1  
This bit is set when the previous TX failed due to an error.
- Bit 10 **ALST1**: Arbitration lost for mailbox1  
This bit is set when the previous TX failed due to an arbitration lost.
- Bit 9 **TXOK1**: Transmission OK of mailbox1  
The hardware updates this bit after each transmission attempt.  
0: The previous transmission failed  
1: The previous transmission was successful  
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Please refer to [Figure 267](#)
- Bit 8 **RQCP1**: Request completed mailbox1  
Set by hardware when the last request (transmit or abort) has been performed.  
Cleared by software writing a "1" or by hardware on transmission request (TXRQ1 set in CAN\_TI1R register).  
Clearing this bit clears all the status bits (TXOK1, ALST1 and TERR1) for Mailbox 1.
- Bit 7 **ABRQ0**: Abort request for mailbox0  
Set by software to abort the transmission request for the corresponding mailbox.  
Cleared by hardware when the mailbox becomes empty.  
Setting this bit has no effect when the mailbox is not pending for transmission.
- Bits 6:4 Reserved, must be kept at reset value.
- Bit 3 **TERR0**: Transmission error of mailbox0  
This bit is set when the previous TX failed due to an error.
- Bit 2 **ALST0**: Arbitration lost for mailbox0  
This bit is set when the previous TX failed due to an arbitration lost.
- Bit 1 **TXOK0**: Transmission OK of mailbox0  
The hardware updates this bit after each transmission attempt.  
0: The previous transmission failed  
1: The previous transmission was successful  
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Please refer to [Figure 267](#)
- Bit 0 **RQCP0**: Request completed mailbox0  
Set by hardware when the last request (transmit or abort) has been performed.  
Cleared by software writing a "1" or by hardware on transmission request (TXRQ0 set in CAN\_TI0R register).  
Clearing this bit clears all the status bits (TXOK0, ALST0 and TERR0) for Mailbox 0.

**CAN receive FIFO 0 register (CAN\_RF0R)**

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	RFOM0	FOVR0	FULL0	Res	FMP0[1:0]										
										rs	rc_w1	rc_w1		r	r

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

Bit 5 **RFOM0**: Release FIFO 0 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.  
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR0**: FIFO 0 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.  
This bit is cleared by software.

Bit 3 **FULL0**: FIFO 0 full

Set by hardware when three messages are stored in the FIFO.  
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP0[1:0]**: FIFO 0 message pending

These bits indicate how many messages are pending in the receive FIFO.  
FMP is increased each time the hardware stores a new message in to the FIFO. FMP is decreased each time the software releases the output mailbox by setting the RFOM0 bit.

**CAN receive FIFO 1 register (CAN\_RF1R)**

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	RFOM1	FOVR1	FULL1	Res	FMP1[1:0]										
										rs	rc_w1	rc_w1		r	r

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

Bit 5 **RFOM1**: Release FIFO 1 output mailbox

Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If at least two messages are pending in the FIFO, the software has to release the output mailbox to access the next message.  
Cleared by hardware when the output mailbox has been released.

Bit 4 **FOVR1**: FIFO 1 overrun

This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.  
This bit is cleared by software.

Bit 3 **FULL1**: FIFO 1 full

Set by hardware when three messages are stored in the FIFO.  
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP1[1:0]**: FIFO 1 message pending

These bits indicate how many messages are pending in the receive FIFO1.  
FMP1 is increased each time the hardware stores a new message in to the FIFO1. FMP is decreased each time the software releases the output mailbox by setting the RFOM1 bit.

### CAN interrupt enable register (CAN\_IER)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SLKIE	WKUIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Res	Res	Res	LEC IE	BOF IE	EPV IE	EWG IE	Res	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **SLKIE**: Sleep interrupt enable

0: No interrupt when SLAKI bit is set.  
1: Interrupt generated when SLAKI bit is set.

Bit 16 **WKUIE**: Wakeup interrupt enable

0: No interrupt when WKUI is set.  
1: Interrupt generated when WKUI bit is set.

Bit 15 **ERRIE**: Error interrupt enable

0: No interrupt will be generated when an error condition is pending in the CAN\_ESR.  
1: An interrupt will be generation when an error condition is pending in the CAN\_ESR.

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

- Bit 11 **LECIE**: Last error code interrupt enable  
0: ERRI bit will not be set when the error code in LEC[2:0] is set by hardware on error detection.  
1: ERRI bit will be set when the error code in LEC[2:0] is set by hardware on error detection.
- Bit 10 **BOFIE**: Bus-off interrupt enable  
0: ERRI bit will not be set when BOFF is set.  
1: ERRI bit will be set when BOFF is set.
- Bit 9 **EPVIE**: Error passive interrupt enable  
0: ERRI bit will not be set when EPVF is set.  
1: ERRI bit will be set when EPVF is set.
- Bit 8 **EWGIE**: Error warning interrupt enable  
0: ERRI bit will not be set when EWGF is set.  
1: ERRI bit will be set when EWGF is set.
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **FOVIE1**: FIFO overrun interrupt enable  
0: No interrupt when FOVR is set.  
1: Interrupt generation when FOVR is set.
- Bit 5 **FFIE1**: FIFO full interrupt enable  
0: No interrupt when FULL bit is set.  
1: Interrupt generated when FULL bit is set.
- Bit 4 **FMPIE1**: FIFO message pending interrupt enable  
0: No interrupt generated when state of FMP[1:0] bits are not 00b.  
1: Interrupt generated when state of FMP[1:0] bits are not 00b.
- Bit 3 **FOVIE0**: FIFO overrun interrupt enable  
0: No interrupt when FOVR bit is set.  
1: Interrupt generated when FOVR bit is set.
- Bit 2 **FFIE0**: FIFO full interrupt enable  
0: No interrupt when FULL bit is set.  
1: Interrupt generated when FULL bit is set.
- Bit 1 **FMPIE0**: FIFO message pending interrupt enable  
0: No interrupt generated when state of FMP[1:0] bits are not 00b.  
1: Interrupt generated when state of FMP[1:0] bits are not 00b.
- Bit 0 **TMEIE**: Transmit mailbox empty interrupt enable  
0: No interrupt when RQCPx bit is set.  
1: Interrupt generated when RQCPx bit is set.
- Note: Refer to [Section 28.8: bxCAN interrupts](#).*

**CAN error status register (CAN\_ESR)**

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
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
Res	Res	Res	Res	Res	Res	Res	Res	Res	LEC[2:0]			Res	BOFF	EPVF	EWGF
									r/w	r/w	r/w		r	r	r

Bits 31:24 **REC[7:0]**: Receive error counter

The implementing part of the fault confinement mechanism of the CAN protocol. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.

Bits 23:16 **TEC[7:0]**: Least significant byte of the 9-bit transmit error counter

The implementing part of the fault confinement mechanism of the CAN protocol.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **LEC[2:0]**: Last error code

This field is set by hardware and holds a code which indicates the error condition of the last error detected on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared to '0'.

The LEC[2:0] bits can be set to value 0b111 by software. They are updated by hardware to indicate the current communication status.

- 000: No Error
- 001: Stuff Error
- 010: Form Error
- 011: Acknowledgment Error
- 100: Bit recessive Error
- 101: Bit dominant Error
- 110: CRC Error
- 111: Set by software

Bit 3 Reserved, must be kept at reset value.

Bit 2 **BOFF**: Bus-off flag

This bit is set by hardware when it enters the bus-off state. The bus-off state is entered on TEC overflow, greater than 255, refer to [Section 28.7.6 on page 777](#).

Bit 1 **EPVF**: Error passive flag

This bit is set by hardware when the Error Passive limit has been reached (Receive Error Counter or Transmit Error Counter > 127).

Bit 0 **EWGF**: Error warning flag

This bit is set by hardware when the warning limit has been reached (Receive Error Counter or Transmit Error Counter ≥ 96).

**CAN bit timing register (CAN\_BTR)**

Address offset: 0x1C

Reset value: 0x0123 0000

This register can only be accessed by the software when the CAN hardware is in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
SILM	LBKM	Res	Res	Res	Res	SJW[1:0]		Res	TS2[2:0]			TS1[3:0]				
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	
Res	Res	Res	Res	Res	Res	BRP[9:0]										
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 31 **SILM**: Silent mode (debug)

- 0: Normal operation
- 1: Silent Mode

Bit 30 **LBKM**: Loop back mode (debug)

- 0: Loop Back Mode disabled
- 1: Loop Back Mode enabled

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **SJW[1:0]**: Resynchronization jump width

These bits define the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform the resynchronization.

$$t_{RJW} = t_{CAN} \times (SJW[1:0] + 1)$$

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **TS2[2:0]**: Time segment 2

These bits define the number of time quanta in Time Segment 2.

$$t_{BS2} = t_{CAN} \times (TS2[2:0] + 1)$$

Bits 19:16 **TS1[3:0]**: Time segment 1

These bits define the number of time quanta in Time Segment 1

$$t_{BS1} = t_{CAN} \times (TS1[3:0] + 1)$$

For more information on bit timing, please refer to [Section 28.7.7: Bit timing on page 777](#).

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

Bits 9:0 **BRP[9:0]**: Baud rate prescaler

These bits define the length of a time quanta.

$$t_q = (BRP[9:0]+1) \times t_{PCLK}$$

### 28.9.3 CAN mailbox registers

This chapter describes the registers of the transmit and receive mailboxes. Refer to [Section 28.7.5: Message storage on page 775](#) for detailed register mapping.

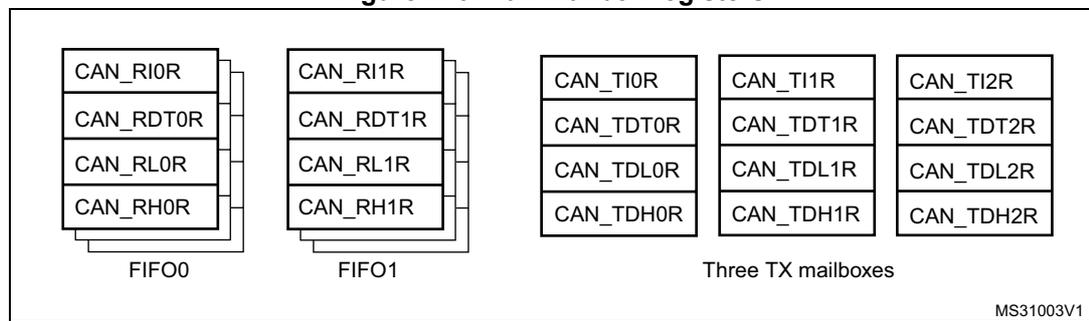
Transmit and receive mailboxes have the same registers except:

- The FMI field in the CAN\_RDTxR register.
- A receive mailbox is always write protected.
- A transmit mailbox is write-enabled only while empty, corresponding TME bit in the CAN\_TSR register set.

There are 3 TX Mailboxes and 2 RX Mailboxes. Each RX Mailbox allows access to a 3 level depth FIFO, the access being offered only to the oldest received message in the FIFO.

Each mailbox consist of 4 registers.

**Figure 276. Can mailbox registers**



**CAN TX mailbox identifier register (CAN\_TlRxR) (x = 0..2)**

Address offsets: 0x180, 0x190, 0x1A0

Reset value: 0xXXXX XXXX (except bit 0, TXRQ = 0)

All TX registers are write protected when the mailbox is pending transmission (TMEx reset).

This register also implements the TX request control (bit 0) - reset value 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
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
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier

The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bit 20:3 **EXID[17:0]**: Extended identifier

The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension

This bit defines the identifier type of message in the mailbox.

0: Standard identifier.

1: Extended identifier.

Bit 1 **RTR**: Remote transmission request

0: Data frame

1: Remote frame

Bit 0 **TXRQ**: Transmit mailbox request

Set by software to request the transmission for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.

**CAN mailbox data length control and time stamp register (CAN\_TDTxR)  
(x = 0..2)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x184, 0x194, 0x1A4

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DLC[3:0]			
												rw	rw	rw	rw

Bits 31:16 **TIME[15:0]**: Message time stamp

This field contains the 16-bit timer value captured at the SOF transmission.

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

Bit 8 **TGT**: Transmit global time

This bit is active only when the hardware is in the Time Trigger Communication mode, TTCM bit of the CAN\_MCR register is set.

0: Time stamp TIME[15:0] is not sent.

1: Time stamp TIME[15:0] value is sent in the last two data bytes of the 8-byte message: TIME[7:0] in data byte 7 and TIME[15:8] in data byte 6, replacing the data written in CAN\_TDHxR[31:16] register (DATA6[7:0] and DATA7[7:0]). DLC must be programmed as 8 in order these two bytes to be sent over the CAN bus.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DLC[3:0]**: Data length code

This field defines the number of data bytes a data frame contains or a remote frame request. A message can contain from 0 to 8 data bytes, depending on the value in the DLC field.

**CAN mailbox data low register (CAN\_TDLxR) (x = 0..2)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x188, 0x198, 0x1A8

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
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
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA3[7:0]**: Data byte 3

Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data byte 2

Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data byte 1

Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data byte 0

Data byte 0 of the message.

A message can contain from 0 to 8 data bytes and starts with byte 0.

**CAN mailbox data high register (CAN\_TDHxR) (x = 0..2)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x18C, 0x19C, 0x1AC

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
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
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- Bits 31:24 **DATA7[7:0]**: Data byte 7  
Data byte 7 of the message.  
*Note: If TGT of this message and TTCM are active, DATA7 and DATA6 will be replaced by the TIME stamp value.*
- Bits 23:16 **DATA6[7:0]**: Data byte 6  
Data byte 6 of the message.
- Bits 15:8 **DATA5[7:0]**: Data byte 5  
Data byte 5 of the message.
- Bits 7:0 **DATA4[7:0]**: Data byte 4  
Data byte 4 of the message.

**CAN receive FIFO mailbox identifier register (CAN\_RlRxR) (x = 0..1)**

Address offsets: 0x1B0, 0x1C0  
Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
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
EXID[12:0]												IDE	RTR	Res	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

- Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier  
The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).
- Bits 20:3 **EXID[17:0]**: Extended identifier  
The LSBs of the extended identifier.
- Bit 2 **IDE**: Identifier extension  
This bit defines the identifier type of message in the mailbox.  
0: Standard identifier.  
1: Extended identifier.
- Bit 1 **RTR**: Remote transmission request  
0: Data frame  
1: Remote frame
- Bit 0 Reserved, must be kept at reset value.



**CAN receive FIFO mailbox data length control and time stamp register (CAN\_RDTxR) (x = 0..1)**

Address offsets: 0x1B4, 0x1C4  
 Reset value: 0xXXXX XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
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
FMI[7:0]								Res	Res	Res	Res	DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:16 **TIME[15:0]**: Message time stamp

This field contains the 16-bit timer value captured at the SOF detection.

Bits 15:8 **FMI[7:0]**: Filter match index

This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering please refer to [Section 28.7.4: Identifier filtering on page 771](#) - **Filter Match Index** paragraph.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DLC[3:0]**: Data length code

This field defines the number of data bytes a data frame contains (0 to 8). It is 0 in the case of a remote frame request.

**CAN receive FIFO mailbox data low register (CAN\_RDLxR) (x = 0..1)**

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x1B8, 0x1C8

Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
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
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA3[7:0]**: Data Byte 3

Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data Byte 2

Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data Byte 1

Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data Byte 0

Data byte 0 of the message.

A message can contain from 0 to 8 data bytes and starts with byte 0.

**CAN receive FIFO mailbox data high register (CAN\_RDHxR) (x = 0..1)**

Address offsets: 0x1BC, 0x1CC

Reset value: 0xFFFF XXXX

All RX registers are write protected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
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
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DATA7[7:0]**: Data Byte 7

Data byte 3 of the message.

Bits 23:16 **DATA6[7:0]**: Data Byte 6

Data byte 2 of the message.

Bits 15:8 **DATA5[7:0]**: Data Byte 5  
 Data byte 1 of the message.

Bits 7:0 **DATA4[7:0]**: Data Byte 4  
 Data byte 0 of the message.

### 28.9.4 CAN filter registers

#### CAN filter master register (CAN\_FMR)

Address offset: 0x200

Reset value: 0x2A1C 0E01

All bits of this register are set and cleared by software.

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

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

Bits 13:8 **CANSB[5:0]**: CAN start bank  
 These bits are set and cleared by software. They define the start bank for the CAN interface (Slave) in the range 1 to 27.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **FINIT**: Filter initialization mode  
 Initialization mode for filter banks  
 0: Active filters mode.  
 1: Initialization mode for the filters.

**CAN filter mode register (CAN\_FM1R)**

Address offset: 0x204

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	FBM27	FBM26	FBM25	FBM24	FBM23	FBM22	FBM21	FBM20	FBM19	FBM18	FBM17	FBM16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBM15	FBM14	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
rw															

*Note:* Please refer to [Figure 269: Filter bank scale configuration - register organization on page 773](#)

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

Bits 27:0 **FBMx**: Filter mode

Mode of the registers of Filter x.

0: Two 32-bit registers of filter bank x are in Identifier Mask mode.

1: Two 32-bit registers of filter bank x are in Identifier List mode.

*Note:* Bits 27:14 are available in connectivity line devices only and are reserved otherwise.

**CAN filter scale register (CAN\_FS1R)**

Address offset: 0x20C

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	FSC27	FSC26	FSC25	FSC24	FSC23	FSC22	FSC21	FSC20	FSC19	FSC18	FSC17	FSC16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSC15	FSC14	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
rw															

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

Bits 27:0 **FSCx**: Filter scale configuration

These bits define the scale configuration of Filters 13-0.

0: Dual 16-bit scale configuration

1: Single 32-bit scale configuration

*Note:* Bits 27:14 are available in connectivity line devices only and are reserved otherwise.

*Note:* Please refer to [Figure 269: Filter bank scale configuration - register organization on page 773](#).

**CAN filter FIFO assignment register (CAN\_FFA1R)**

Address offset: 0x214

Reset value: 0x0000 0000

This register can be written only when the filter initialization mode is set (FINIT=1) in the CAN\_FMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	FFA27	FFA26	FFA25	FFA24	FFA23	FFA22	FFA21	FFA20	FFA19	FFA18	FFA17	FFA16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFA15	FFA14	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
rw															

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

Bits 27:0 **FFAx**: Filter FIFO assignment for filter x

The message passing through this filter will be stored in the specified FIFO.

0: Filter assigned to FIFO 0

1: Filter assigned to FIFO 1

*Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.*

**CAN filter activation register (CAN\_FA1R)**

Address offset: 0x21C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	FACT27	FACT26	FACT25	FACT24	FACT23	FACT22	FACT21	FACT20	FACT19	FACT18	FACT17	FACT16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FACT15	FACT14	FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
rw															

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

Bits 27:0 **FACTx**: Filter active

The software sets this bit to activate Filter x. To modify the Filter x registers (CAN\_FxR[0:7]), the FACTx bit must be cleared or the FINIT bit of the CAN\_FMR register must be set.

0: Filter x is not active

1: Filter x is active

*Note: Bits 27:14 are available in connectivity line devices only and are reserved otherwise.*

**Filter bank i register x (CAN\_FiRx) (i = 0..27, x = 1, 2)**

Address offsets: 0x240..0x31C

Reset value: 0xFFFF XXXX

There are 28 filter banks, i=0 .. 27. Each filter bank i is composed of two 32-bit registers, CAN\_FiR[2:1].

This register can only be modified when the FACTx bit of the CAN\_FAxR register is cleared or when the FINIT bit of the CAN\_FMR register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
r/w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
r/w															

In all configurations:

Bits 31:0 **FB[31:0]**: Filter bits

**Identifier**

Each bit of the register specifies the level of the corresponding bit of the expected identifier.

0: Dominant bit is expected

1: Recessive bit is expected

**Mask**

Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier or not.

0: Don't care, the bit is not used for the comparison

1: Must match, the bit of the incoming identifier must have the same level has specified in the corresponding identifier register of the filter.

*Note:* Depending on the scale and mode configuration of the filter the function of each register can differ. For the filter mapping, functions description and mask registers association, refer to [Section 28.7.4: Identifier filtering on page 771](#).

A Mask/Identifier register in **mask mode** has the same bit mapping as in **identifier list mode**.

For the register mapping/addresses of the filter banks please refer to the [Table 105 on page 803](#).



Table 105. bxCAN 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
0x184	CAN_TDT0R	TIME[15:0]															Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]										
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x188	CAN_TDL0R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	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	x	x	x	x
0x18C	CAN_TDH0R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	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	x	x	x	x
0x190	CAN_TI1R	STID[10:0]/EXID[28:18]											EXID[17:0]															IDE	RTR	TXRQ			
	Reset value	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	x	x	x	0
0x194	CAN_TDT1R	TIME[15:0]															Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]									
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x198	CAN_TDL1R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	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	x	x	x	x
0x19C	CAN_TDH1R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	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	x	x	x	x
0x1A0	CAN_TI2R	STID[10:0]/EXID[28:18]											EXID[17:0]															IDE	RTR	TXRQ			
	Reset value	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	x	x	x	0
0x1A4	CAN_TDT2R	TIME[15:0]															Res.	Res.	TGT	Res.	Res.	Res.	Res.	DLC[3:0]									
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x1A8	CAN_TDL2R	DATA3[7:0]							DATA2[7:0]							DATA1[7:0]							DATA0[7:0]										
	Reset value	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	x	x	x	x
0x1AC	CAN_TDH2R	DATA7[7:0]							DATA6[7:0]							DATA5[7:0]							DATA4[7:0]										
	Reset value	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	x	x	x	x
0x1B0	CAN_RI0R	STID[10:0]/EXID[28:18]											EXID[17:0]															IDE	RTR	Res.			
	Reset value	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	x	x	x	





Table 105. bxCAN 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				
0x218	Reserved																																				
0x21C	CAN_FA1R	Res.	Res.	Res.	Res.	FACT[27:0]																															
	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			
0x220	Reserved																																				
0x224-0x23F	Reserved																																				
0x240	CAN_F0R1	FB[31:0]																																			
	Reset value	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	x	x	x	x				
0x244	CAN_F0R2	FB[31:0]																																			
	Reset value	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	x	x	x	x				
0x248	CAN_F1R1	FB[31:0]																																			
	Reset value	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	x	x	x	x				
0x24C	CAN_F1R2	FB[31:0]																																			
	Reset value	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	x	x	x	x				
⋮	⋮	⋮																																			
0x318	CAN_F27R1	FB[31:0]																																			
	Reset value	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	x	x	x	x				
0x31C	CAN_F27R2	FB[31:0]																																			
	Reset value	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	x	x	x	x				



## 29 Universal serial bus full-speed device interface (USB)

### 29.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported which allows to stop the device clocks for low-power consumption.

### 29.2 USB main features

- USB specification version 2.0 full-speed compliant
- Configurable number of endpoints from 1 to 8
- 512 bytes of dedicated packet buffer memory SRAM
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint support
- USB Suspend/Resume operations
- Frame locked clock pulse generation

### 29.3 USB implementation

[Table 106](#) describes the USB implementation in STM32F37xx devices.

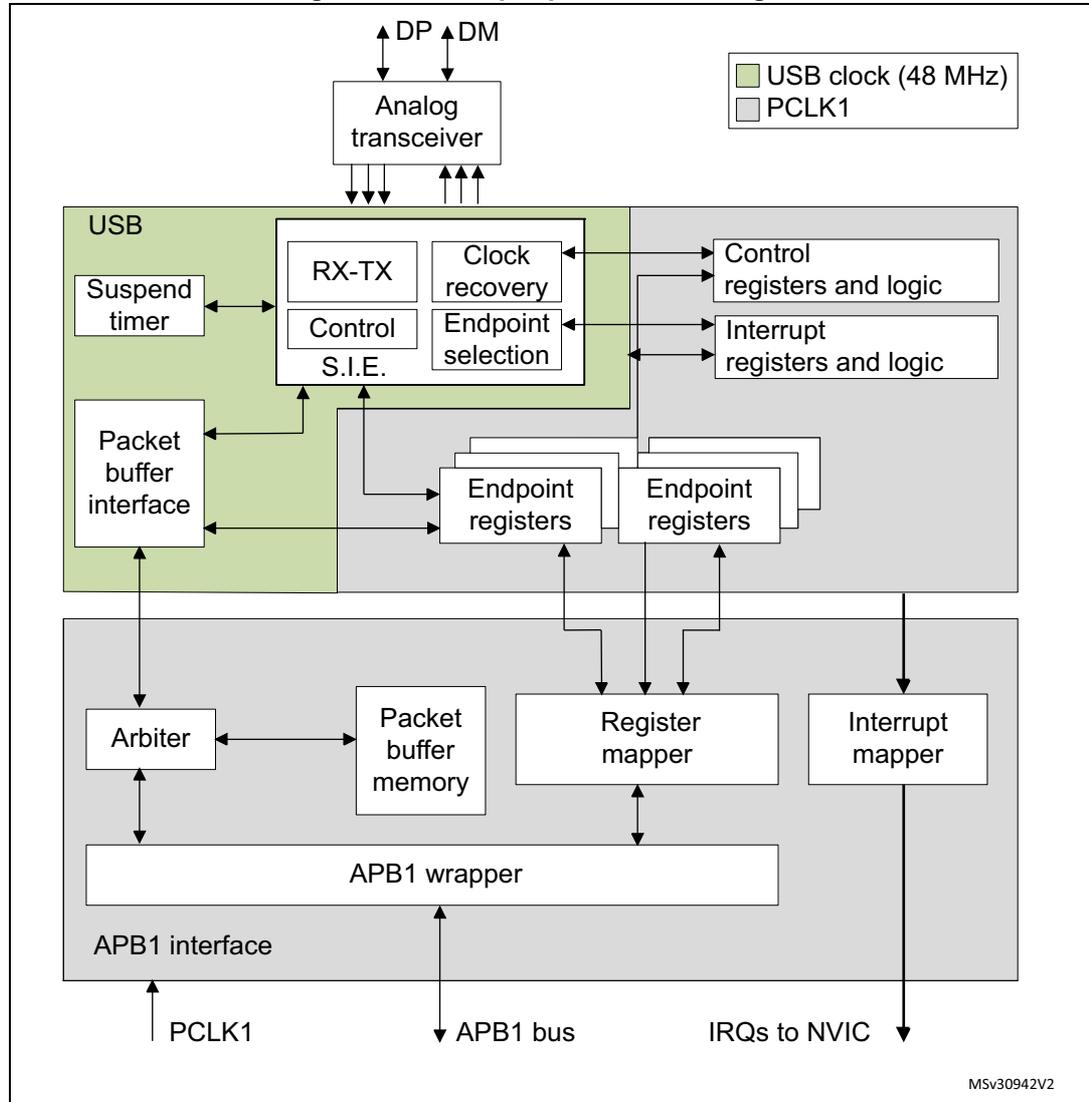
**Table 106. STM32F37xx USB implementation**

USB features	USB
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	512 bytes
Dedicated packet buffer memory SRAM access scheme	2 x 16 bits / word

## 29.4 USB functional description

Figure 277 shows the block diagram of the USB peripheral.

Figure 277. USB peripheral block diagram



The USB peripheral provides an USB-compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is up to 512 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the USB host, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a buffer description block indicating where the endpoint-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes place. The data

buffered by the USB peripheral is loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which allows to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wakeup line to allow the system to immediately restart the normal clock generation and/or support direct clock start/stop.

### 29.4.1 Description of USB blocks

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- **Serial Interface Engine (SIE):** The functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage,. This unit also generates signals according to USB peripheral events, such as Start of Frame (SOF), USB\_Reset, Data errors etc. and to Endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.
- **Timer:** This block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- **Packet Buffer Interface:** This block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the Endpoint registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- **Endpoint-Related Registers:** Each endpoint has an associated register containing the endpoint type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer

endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.

- Control Registers: These are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt Registers: These contain the Interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note:* \* Endpoint 0 is always used for control transfer in single-buffer mode.

The USB peripheral is connected to the APB1 bus through an APB1 interface, containing the following blocks:

- Packet Memory: This is the local memory that physically contains the Packet Buffers. It can be used by the Packet Buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the Packet Memory is up to 512 bytes, structured as 256 half-words by 16 bits.
- Arbiter: This block accepts memory requests coming from the APB1 bus and from the USB interface. It resolves the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.
- Register Mapper: This block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 16-bit wide half-word set addressed by the APB1.
- APB1 Wrapper: This provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.
- Interrupt Mapper: This block is used to select how the possible USB events can generate interrupts and map them to three different lines of the NVIC:
  - USB low-priority interrupt (Channel 20): Triggered by all USB events (Correct transfer, USB reset, etc.). The firmware has to check the interrupt source before serving the interrupt.
  - USB high-priority interrupt (Channel 19): Triggered only by a correct transfer event for isochronous and double-buffer bulk transfer to reach the highest possible transfer rate.
  - USB wakeup interrupt (Channel 42): Triggered by the wakeup event from the USB Suspend mode.

## 29.5 Programming considerations

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

### 29.5.1 Generic USB device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and Isochronous transfers. Apart from system reset, action is always initiated by the USB peripheral, driven by one of the USB events described below.

## 29.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software should perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ( $t_{\text{STARTUP}}$  specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the FRES bit in the CNTR register). Clearing the ISTR register then removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

### USB reset (RESET interrupt)

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral will not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the Enable Function (EF) bit of the USB\_DADDR register and initializing the EP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB\_DADDR register, and configures any other necessary endpoint.

When a RESET interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of reset sequence which triggered the interrupt.

### Structure and usage of packet buffers

Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has to be accessed by the microcontroller also, an arbitration logic takes care of the access conflicts, using half APB1 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both the agents can operate as if the packet memory is a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data will be copied to the memory only up to the last available location.

### Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn\_TX/ADDRn\_RX registers so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The EP\_TYPE bits in the USB\_EPnR register must be set according to the endpoint type, eventually using the EP\_KIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STAT\_TX bits in the USB\_EPnR register and COUNTn\_TX must be initialized. For reception, STAT\_RX bits must be set to enable reception and COUNTn\_RX must be written with the allocated buffer size using the BL\_SIZE and NUM\_BLOCK fields. Unidirectional endpoints, except Isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB\_EPnR and locations ADDRn\_TX/ADDRn\_RX, COUNTn\_TX/COUNTn\_RX (respectively), should not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

### IN packets (data transmission)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of ADDRn\_TX and COUNTn\_TX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16 bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (Refer to [Structure and usage of packet buffers on page 811](#)) and starts sending a DATA0 or DATA1 PID according to USB\_EPnR bit DTOG\_TX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STAT\_TX bits in the USB\_EPnR register.

The ADDR internal register is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn\_TX for COUNTn\_TX/2 half-words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed will be used.

On receiving the ACK receipt by the host, the USB\_EPnR register is updated in the following way: DTOG\_TX bit is toggled, the endpoint is made invalid by setting STAT\_TX=10 (NAK) and bit CTR\_TX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP\_ID and DIR bits in the USB\_ISTR register. Servicing of the CTR\_TX event starts clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn\_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STAT\_TX to '11 (VALID) to re-enable transmissions. While the STAT\_TX bits are equal to '10 (NAK), any IN request addressed to that endpoint is NAKed,

indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

### **OUT and SETUP packets (data reception)**

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn\_RX and COUNTn\_RX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn\_RX is stored directly in its internal register ADDR. While COUNT is now reset and the values of BL\_SIZE and NUM\_BLOCK bit fields, which are read within COUNTn\_RX content are used to initialize BUF\_COUNT, an internal 16 bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF\_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but ACK packet is not sent and the ERR bit in USB\_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STAT\_RX in the USB\_EPnR register and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn\_RX for a number of bytes corresponding to the received data packet length, CRC included (i.e. data payload length + 2), or up to the last allocated memory location, as defined by BL\_SIZE and NUM\_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn\_RX location inside the buffer description table entry, leaving unaffected BL\_SIZE and NUM\_BLOCK fields, which normally do not require to be re-written, and the USB\_EPnR register is updated in the following way: DTOG\_RX bit is toggled, the endpoint is made invalid by setting STAT\_RX = '10 (NAK) and bit CTR\_RX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP\_ID and DIR bits in the USB\_ISTR register. The CTR\_RX event is serviced by first determining the transaction type (SETUP bit in the USB\_EPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn\_RX location inside the buffer description table entry related to the endpoint being

processed. After the received data is processed, the application software should set the STAT\_RX bits to '11 (Valid) in the USB\_EPnR, enabling further transactions. While the STAT\_RX bits are equal to '10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

### Control transfers

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOG\_TX and DTOG\_RX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STAT\_TX and STAT\_RX are set to '10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB\_EPnR register at each CTR\_RX event to distinguish normal OUT transactions from SETUP ones. A USB device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction should be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction should be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software will change NAK to VALID, otherwise to STALL. At the same time, if the status stage will be an OUT, the STATUS\_OUT (EP\_KIND in the USB\_EPnR register) bit should be set, so that an error is generated if a status transaction is performed with not-zero data. When the status transaction is serviced, the application clears the STATUS\_OUT bit and sets STAT\_RX to VALID (to accept a new command) and STAT\_TX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic doesn't allow a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STAT\_RX bits are set to '01 (STALL) or '10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued CTR\_RX request not yet acknowledged by the application (i.e. CTR\_RX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the CTR\_RX interrupt.

### 29.5.3 Double-buffered endpoints

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the

bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it will answer with a NAK handshake and the host PC will issue the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB\_EPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from ‘00 (Disabled): STAT\_RX if the double-buffered bulk endpoint is enabled for reception, STAT\_TX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB\_EPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOG\_RX (bit 14 of USB\_EPnR register) for ‘reception’ double-buffered bulk endpoints or DTOG\_TX (bit 6 of USB\_EPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB\_EPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW\_BUF. In the following table the correspondence between USB\_EPnR register bits and DTOG/SW\_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.

**Table 107. Double-buffering buffer flag definition**

Buffer flag	‘Transmission’ endpoint	‘Reception’ endpoint
DTOG	DTOG_TX (USB_EPnR bit 6)	DTOG_RX (USB_EPnR bit 14)
SW_BUF	USB_EPnR bit 14	USB_EPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW\_BUF

buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

**Table 108. Bulk double-buffering memory buffers usage**

Endpoint Type	DTOG	SW_BUF	Packet buffer used by USB Peripheral	Packet buffer used by Application Software
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.
	1	0	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	0	0	None <sup>(1)</sup>	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	1	1	None <sup>(1)</sup>	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.
	1	0	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	0	0	None <sup>(1)</sup>	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	1	1	None <sup>(1)</sup>	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by:

- Writing EP\_TYPE bit field at '00 in its USB\_EPnR register, to define the endpoint as a bulk, and
- Setting EP\_KIND bit at '1 (DBL\_BUF), in the same register.

The application software is responsible for DTOG and SW\_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL\_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL\_BUF remain set. At the end of each transaction the CTR\_RX or CTR\_TX bit of the addressed endpoint USB\_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB\_EPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL\_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains '11 (Valid). However, as the token packet of a new transaction is received, the actual endpoint status will be masked as '10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW\_BUF having the same value, see [Table 108 on page 817](#)). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW\_BUF bit, writing '1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by

the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control will take place and the actual transfer rate will be limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from '11 (Valid) into the STAT bit pair of the related USB\_EPnR register. In this case, the USB peripheral will always use the programmed endpoint status, regardless of the buffer usage condition.

### 29.5.4 Isochronous transfers

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as 'Isochronous'. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be 'isochronous' during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for Isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, Isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The Isochronous behavior for an endpoint is selected by setting the EP\_TYPE bits at '10 in its USB\_EPnR register; since there is no handshake phase the only legal values for the STAT\_RX/STAT\_TX bit pairs are '00 (Disabled) and '11 (Valid), any other value will produce results not compliant to USB standard. Isochronous endpoints implement double-buffering to ease application software development, using both 'transmission' and 'reception' packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOG\_RX for 'reception' isochronous endpoints, DTOG\_TX for 'transmission' isochronous endpoints, both in the related USB\_EPnR register) according to [Table 109](#).

**Table 109. Isochronous memory buffers usage**

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
IN	0	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.
	1	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.
OUT	0	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.
	1	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB\_EPnR registers used to implement Isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have Isochronous endpoints enabled both for reception and transmission, two USB\_EPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the CTR\_RX or CTR\_TX bit of the addressed endpoint USB\_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB\_EPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for Isochronous transfers due to the lack of handshake phase, the endpoint remains always '11 (Valid). CRC errors or buffer-overflow conditions occurring during Isochronous OUT transfers are anyway considered as correct transactions and they always trigger an CTR\_RX event. However, CRC errors will anyway set the ERR bit in the USB\_ISTR register to notify the software of the possible data corruption.

### 29.5.5 Suspend/Resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to '1 in USB\_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the FSUSP bit in the USB\_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set LP\_MODE bit in USB\_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (See “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the LP\_MODE bit in USB\_CNTR register asynchronously. Even if this event can trigger an WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure should address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear FSUSP bit of USB\_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB\_FNR register can be used according to [Table 110](#), which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the Idle bus state; moreover at the end of a reset sequence the RESET bit in USB\_ISTR register is set to 1, issuing an interrupt if enabled, which should be handled as usual.

**Table 110. Resume event detection**

[RXDP,RXDM] status	Wakeup event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (e.g. a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the RESUME bit in the USB\_CNTR register to ‘1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the RESUME bit is clear, the resume sequence will be completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB\_FNR register.

*Note:* The RESUME bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the FSUSP bit in USB\_CNTR register to 1.

## 29.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common Registers: Interrupt and Control registers
- Endpoint Registers: Endpoint configuration and status
- Buffer Descriptor Table: Location of packet memory used to locate data buffers

All register addresses are expressed as offsets with respect to the USB peripheral registers base address 0x4000 5C00, except the buffer descriptor table locations, which starts at the address specified by the USB\_BTABLE register. All register addresses are aligned to 32-bit word boundaries although they are 16-bit wide. The same address alignment is used to access packet buffer memory locations, which are located starting from 0x4000 6000.

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

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

### 29.6.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

#### USB control register (USB\_CNTR)

Address offset: 0x40

Reset value: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Res.	Res.	Res.	RESUME	FSUSP	LP_MODE	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

- Bit 15 **CTRM**: Correct transfer interrupt mask
  - 0: Correct Transfer (CTR) Interrupt disabled.
  - 1: CTR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 14 **PMAOVRM**: Packet memory area over / underrun interrupt mask
  - 0: PMAOVR Interrupt disabled.
  - 1: PMAOVR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 13 **ERRM**: Error interrupt mask
  - 0: ERR Interrupt disabled.
  - 1: ERR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 12 **WKUPM**: Wakeup interrupt mask
  - 0: WKUP Interrupt disabled.
  - 1: WKUP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.

- Bit 11 **SUSPM**: Suspend mode interrupt mask  
0: Suspend Mode Request (SUSP) Interrupt disabled.  
1: SUSP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 10 **RESETM**: USB reset interrupt mask  
0: RESET Interrupt disabled.  
1: RESET Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 9 **SOFM**: Start of frame interrupt mask  
0: SOF Interrupt disabled.  
1: SOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 8 **ESOFM**: Expected start of frame interrupt mask  
0: Expected Start of Frame (ESOF) Interrupt disabled.  
1: ESOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bits 7:5 Reserved.
- Bit 4 **RESUME**: Resume request  
The microcontroller can set this bit to send a Resume signal to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the Host PC is ready to drive the resume sequence up to its end.
- Bit 3 **FSUSP**: Force suspend  
Software must set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms.  
0: No effect.  
1: Enter suspend mode. Clocks and static power dissipation in the analog transceiver are left unaffected. If suspend power consumption is a requirement (bus-powered device), the application software should set the LP\_MODE bit after FSUSP as explained below.
- Bit 2 **LP\_MODE**: Low-power mode  
This mode is used when the suspend-mode power constraints require that all static power dissipation is avoided, except the one required to supply the external pull-up resistor. This condition should be entered when the application is ready to stop all system clocks, or reduce their frequency in order to meet the power consumption requirements of the USB suspend condition. The USB activity during the suspend mode (WKUP event) asynchronously resets this bit (it can also be reset by software).  
0: No Low-power mode.  
1: Enter Low-power mode.
- Bit 1 **PDWN**: Power down  
This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.  
0: Exit Power Down.  
1: Enter Power down mode.
- Bit 0 **FRES**: Force USB Reset  
0: Clear USB reset.  
1: Force a reset of the USB peripheral, exactly like a RESET signalling on the USB. The USB peripheral is held in RESET state until software clears this bit. A "USB-RESET" interrupt is generated, if enabled.

**USB interrupt status register (USB\_ISTR)**

Address offset: 0x44

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Res.	Res.	Res.	DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r	r	r	r

This register contains the status of all the interrupt sources allowing application software to determine, which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB\_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, will perform all necessary actions, and finally it will clear the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line will be kept high again. If several bits are set simultaneously, only a single interrupt will be generated.

Endpoint transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB\_CNTR is set. An endpoint dedicated interrupt condition is activated independently from the CTRM bit in the USB\_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB\_EPnR register (the CTR bit is actually a read only bit). For endpoint-related interrupts, the software can use the Direction of Transaction (DIR) and EP\_ID read-only bits to identify, which endpoint made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB\_ISTR events by specifying the order in which software checks USB\_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt will be requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with '0 (these bits can only be cleared by software). Read-modify-write cycles should be avoided because between the read and the write operations another bit could be set by the hardware and the next write will clear it before the microprocessor has the time to serve the event.

The following describes each bit in detail:

Bit 15 **CTR**: Correct transfer

This bit is set by the hardware to indicate that an endpoint has successfully completed a transaction; using DIR and EP\_ID bits software can determine which endpoint requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR**: Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host will retry the transaction. The PMAOVR interrupt should never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during Isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 13 **ERR**: Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic Redundancy Check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit Stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format Violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (e.g. loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 12 **WKUP**: Wakeup

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the LP\_MODE bit in the CTLR register and activates the USB\_WAKEUP line, which can be used to notify the rest of the device (e.g. wakeup unit) about the start of the resume process. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bit 11 **SUSP**: Suspend mode request

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (FSUSP=1) until the end of resume sequence. This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 10 RESET:** USB reset request

Set when the USB peripheral detects an active USB RESET signal at its inputs. The USB peripheral, in response to a RESET, just resets its internal protocol state machine, generating an interrupt if RESETM enable bit in the USB\_CNTR register is set. Reception and transmission are disabled until the RESET bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RESET interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 9 SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB\_FNR register which is updated at the SOF packet reception (this could be useful for isochronous applications). This bit is read/write but only '0 can be written and writing '1 has no effect.

**Bit 8 ESOF:** Expected start of frame

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the hub does not receive it properly, the Suspend Timer issues this interrupt. If three consecutive ESOF interrupts are generated (i.e. three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the Suspend Timer is not yet locked. This bit is read/write but only '0 can be written and writing '1 has no effect.

Bits 7:5 Reserved.

**Bit 4 DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.

If DIR bit=0, CTR\_TX bit is set in the USB\_EPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).

If DIR bit=1, CTR\_RX bit or both CTR\_TX/CTR\_RX are set in the USB\_EPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed.

This information can be used by the application software to access the USB\_EPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

**Bits 3:0 EP\_ID[3:0]:** Endpoint Identifier

These bits are written by the hardware according to the endpoint number, which generated the interrupt request. If several endpoint transactions are pending, the hardware writes the endpoint identifier related to the endpoint having the highest priority defined in the following way: Two endpoint sets are defined, in order of priority: Isochronous and double-buffered bulk endpoints are considered first and then the other endpoints are examined. If more than one endpoint from the same set is requesting an interrupt, the EP\_ID bits in USB\_ISTR register are assigned according to the lowest requesting endpoint register, EP0R having the highest priority followed by EP1R and so on. The application software can assign a register to each endpoint according to this priority scheme, so as to order the concurring endpoint requests in a suitable way. These bits are read only.

**USB frame number register (USB\_FNR)**

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 15 **RXDP**: Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 14 **RXDM**: Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 13 **LCK**: Locked

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]**: Lost SOF

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]**: Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for Isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

**USB device address (USB\_DADDR)**

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved

Bit 7 **EF**: Enable function

This bit is set by the software to enable the USB device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at '0 no transactions are handled, irrespective of the settings of USB\_EPnR registers.

Bits 6:0 **ADD[6:0]**: Device address

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the Endpoint Address (EA) field in the associated USB\_EPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

**Buffer table address (USB\_BTABLE)**

Address offset: 0x50

Reset value: 0x0000

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

Bits 15:3 **BTABLE[15:3]**: Buffer table

These bits contain the start address of the buffer allocation table inside the dedicated packet memory. This table describes each endpoint buffer location and size and it must be aligned to an 8 byte boundary (the 3 least significant bits are always '0'). At the beginning of every transaction addressed to this device, the USB peripheral reads the element of this table related to the addressed endpoint, to get its buffer start location and the buffer size (Refer to [Structure and usage of packet buffers on page 811](#)).

Bits 2:0 Reserved, forced by hardware to 0.

**29.6.2 Endpoint-specific registers**

The number of these registers varies according to the number of endpoints that the USB peripheral is designed to handle. The USB peripheral supports up to 8 bidirectional endpoints. Each USB device must support a control endpoint whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint number value. For each endpoint, an USB\_EPnR register is available to store the endpoint specific information.

**USB endpoint n register (USB\_EPnR), n=[0..7]**

Address offset: 0x00 to 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP_TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

They are also reset when an USB reset is received from the USB bus or forced through bit FRES in the CTLR register, except the CTR\_RX and CTR\_TX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint has its USB\_EPnR register where n is the endpoint identifier.

Read-modify-write cycles on these registers should be avoided because between the read and the write operations some bits could be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an 'invariant' value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their 'invariant' value.

**Bit 15 CTR\_RX:** Correct Transfer for reception

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB\_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only '0' can be written, writing 1 has no effect.

**Bit 14 DTOG\_RX:** Data Toggle, for reception transfers

If the endpoint is not Isochronous, this bit contains the expected value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent to the USB host, following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to [Section 29.5.3: Double-buffered endpoints](#)).

If the endpoint is Isochronous, this bit is used only to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to [Section 29.5.4: Isochronous transfers](#)). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes '0', the value of DTOG\_RX remains unchanged, while writing '1' makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STAT\_RX [1:0]**: Status bits, for reception transfers

These bits contain information about the endpoint status, which are listed in [Table 111: Reception status encoding on page 830](#). These bits can be toggled by software to initialize their value. When the application software writes '0, the value remains unchanged, while writing '1 makes the bit value toggle. Hardware sets the STAT\_RX bits to NAK when a correct transfer has occurred (CTR\_RX=1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledge a new transaction

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to [Section 29.5.3: Double-buffered endpoints](#)).

If the endpoint is defined as Isochronous, its status can be only "VALID" or "DISABLED", so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT\_RX bits to 'STALL' or 'NAK' for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing '1.

Bit 11 **SETUP**: Setup transaction completed

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (CTR\_RX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while CTR\_RX bit is at 1; its state changes when CTR\_RX is at 0. This bit is read-only.

Bits 10:9 **EP\_TYPE[1:0]**: Endpoint type

These bits configure the behavior of this endpoint as described in [Table 112: Endpoint type encoding on page 831](#). Endpoint 0 must always be a control endpoint and each USB function must have at least one control endpoint which has address 0, but there may be other control endpoints if required. Only control endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK, the USB peripheral will not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint is defined as STALL in the receive direction, then the SETUP packet will be accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint is a control one.

Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EP\_KIND configuration bit.

The usage of Isochronous endpoints is explained in [Section 29.5.4: Isochronous transfers](#)

Bit 8 **EP\_KIND**: Endpoint kind

The meaning of this bit depends on the endpoint type configured by the EP\_TYPE bits. [Table 113](#) summarizes the different meanings.

DBL\_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in [Section 29.5.3: Double-buffered endpoints](#).

STATUS\_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered 'STALL' instead of 'ACK'. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS\_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **CTR\_TX**: Correct Transfer for transmission

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB\_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only '0 can be written.

Bit 6 **DTOG\_TX**: Data Toggle, for transmission transfers

If the endpoint is non-isochronous, this bit contains the required value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to [Section 29.5.3: Double-buffered endpoints](#))

If the endpoint is Isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to [Section 29.5.4: Isochronous transfers](#)). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for Isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes '0, the value of DTOG\_TX remains unchanged, while writing '1 makes the bit value toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STAT\_TX [1:0]**: Status bits, for transmission transfers

These bits contain the information about the endpoint status, listed in [Table 114](#). These bits can be toggled by the software to initialize their value. When the application software writes '0, the value remains unchanged, while writing '1 makes the bit value toggle. Hardware sets the STAT\_TX bits to NAK, when a correct transfer has occurred (CTR\_TX=1) corresponding to a IN or SETUP (control only) transaction addressed to this endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to [Section 29.5.3: Double-buffered endpoints](#)).

If the endpoint is defined as Isochronous, its status can only be "VALID" or "DISABLED".

Therefore, the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT\_TX bits to 'STALL' or 'NAK' for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing '1.

Bits 3:0 **EA[3:0]**: Endpoint address

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

**Table 111. Reception status encoding**

STAT_RX[1:0]	Meaning
00	<b>DISABLED</b> : all reception requests addressed to this endpoint are ignored.
01	<b>STALL</b> : the endpoint is stalled and all reception requests result in a STALL handshake.
10	<b>NAK</b> : the endpoint is naked and all reception requests result in a NAK handshake.
11	<b>VALID</b> : this endpoint is enabled for reception.

**Table 112. Endpoint type encoding**

EP_TYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

**Table 113. Endpoint kind meaning**

EP_TYPE[1:0]		EP_KIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	Not used
11	INTERRUPT	Not used

**Table 114. Transmission status encoding**

STAT_TX[1:0]	Meaning
00	<b>DISABLED:</b> all transmission requests addressed to this endpoint are ignored.
01	<b>STALL:</b> the endpoint is stalled and all transmission requests result in a STALL handshake.
10	<b>NAK:</b> the endpoint is naked and all transmission requests result in a NAK handshake.
11	<b>VALID:</b> this endpoint is enabled for transmission.

### 29.6.3 Buffer descriptor table

Although the buffer descriptor table is located inside the packet buffer memory, its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

All packet memory locations are accessed by the APB using 32-bit aligned addresses, instead of the actual memory location addresses utilized by the USB peripheral for the USB\_BTABLE register and buffer description table locations.

In the following pages two location addresses are reported: the one to be used by application software while accessing the packet memory, and the local one relative to USB Peripheral access. To obtain the correct memory address value to be used in the application software while accessing the packet memory, the actual memory location address must be multiplied by two. The first packet memory location is located at 0x4000 6000. The buffer descriptor table entry associated with the USB\_EPnR registers is described below.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in [Structure and usage of packet buffers on page 811](#).

#### Transmission buffer address n (USB\_ADDRn\_TX)

Address offset: [USB\_BTABLE] + n\*16

USB local address: [USB\_BTABLE] + n\*8

*Note: In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB\_ADDRn\_TX\_0.*

*In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB\_ADDRn\_RX\_0.*

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

Bits 15:1 **ADDRn\_TX[15:1]**: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint associated with the USB\_EPnR register at the next IN token addressed to it.

Bit 0 Must always be written as '0 since packet memory is half-word wide and all packet buffers must be half-word aligned.

**Transmission byte count n (USB\_COUNTn\_TX)**

Address offset: [USB\_BTABLE] + n\*16 + 4

USB local Address: [USB\_BTABLE] + n\*8 + 2

*Note: In case of double-buffered or isochronous endpoints in the IN direction, this address location is referred to as USB\_COUNTn\_TX\_0.*

*In case of double-buffered or isochronous endpoints in the OUT direction, this address location is used for USB\_COUNTn\_RX\_0.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	COUNTn_TX[9:0]									
						r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:10 These bits are not used since packet size is limited by USB specifications to 1023 bytes. Their value is not considered by the USB peripheral.

Bits 9:0 **COUNTn\_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint associated with the USB\_EPnR register at the next IN token addressed to it.

**Reception buffer address n (USB\_ADDRn\_RX)**

Address offset: [USB\_BTABLE] + n\*16 + 8

USB local Address: [USB\_BTABLE] + n\*8 + 4

*Note: In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB\_ADDRn\_RX\_1.*

*In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB\_ADDRn\_TX\_1.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ADDRn_RX[15:1]															-	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	-

Bits 15:1 **ADDRn\_RX[15:1]:** Reception buffer address

These bits point to the starting address of the packet buffer, which will contain the data received by the endpoint associated with the USB\_EPnR register at the next OUT/SETUP token addressed to it.

Bit 0 This bit must always be written as '0 since packet memory is half-word wide and all packet buffers must be half-word aligned.

**Reception byte count n (USB\_COUNTn\_RX)**

Address offset: [USB\_BTABLE] + n\*16 + 12

USB local Address: [USB\_BTABLE] + n\*8 + 6

*Note: In case of double-buffered or isochronous endpoints in the OUT direction, this address location is referred to as USB\_COUNTn\_RX\_1.*

*In case of double-buffered or isochronous endpoints in the IN direction, this address location is used for USB\_COUNTn\_TX\_1.*

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

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (See “Universal Serial Bus Specification”).

**Bit 15 BL\_SIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BL\_SIZE=0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BL\_SIZE=1, the memory block is 32-byte large, which allows to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

**Bits 14:10 NUM\_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BL\_SIZE value as illustrated in [Table 115](#).

**Bits 9:0 COUNTn\_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint associated with the USB\_EPnR register during the last OUT/SETUP transaction addressed to it.

Table 115. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BL_SIZE=0	Memory allocated when BL_SIZE=1
0 ('00000)	Not allowed	32 bytes
1 ('00001)	2 bytes	64 bytes
2 ('00010)	4 bytes	96 bytes
3 ('00011)	6 bytes	128 bytes
...	...	...
14 ('01110)	28 bytes	480 bytes
15 ('01111)	30 bytes	N/A
16 ('10000)	32 bytes	N/A
...	...	...
29 ('11101)	58 bytes	N/A
30 ('11110)	60 bytes	N/A
31 ('11111)	62 bytes	N/A

### 29.6.4 USB register map

The table below provides the USB register map and reset values.

Table 116. USB 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	USB_EP0R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	USB_EP1R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	USB_EP2R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	USB_EP3R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	USB_EP4R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	



Table 116. USB 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		
0x14	USB_EP5R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x18	USB_EP6R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	USB_EP7R	Res.	CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]																							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20-0x3F	Reserved																																		
0x40	USB_CNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	USB_ISTR	Res.	CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Res.	Res.	Res.	DIR	EP_ID[3:0]																				
	Reset value																		0	0	0	0	0	0	0	0				0	0	0	0	0	0
0x48	USB_FNR	Res.	RXDP	RXDM	LCK	LSOF [1:0]	FN[10:0]																												
	Reset value																		0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	
0x4C	USB_DADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]																								
	Reset value																									0	0	0	0	0	0	0	0	0	0
0x50	USB_BTABLE	Res.	BTABLE[15:3]										Res.	Res.	Res.																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to [Section 2.2.2](#) for the register boundary addresses.

## 30 HDMI-CEC controller (HDMI-CEC)

### 30.1 Introduction

Consumer Electronics Control (CEC) is part of HDMI (High-Definition Multimedia Interface) standard as *appendix supplement 1*.

It consists of a protocol that provides high-level control functions between all of the various audiovisual products in a user environment. It has been specified to operate at low speeds with minimal processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

### 30.2 HDMI-CEC controller main features

- Complies with HDMI-CEC v1.4 Specification
- 32 KHz CEC kernel with 2 clock source options
  - HSI RC oscillator with fixed prescaler (HSI/244)
  - LSE oscillator
- Works in Stop mode for ultra low-power applications
- Configurable Signal Free Time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable Peripheral Address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-Error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration Lost detection (ARBLST)
  - With automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)

### 30.3 HDMI-CEC functional description

#### 30.3.1 HDMI-CEC pin

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-and connection.

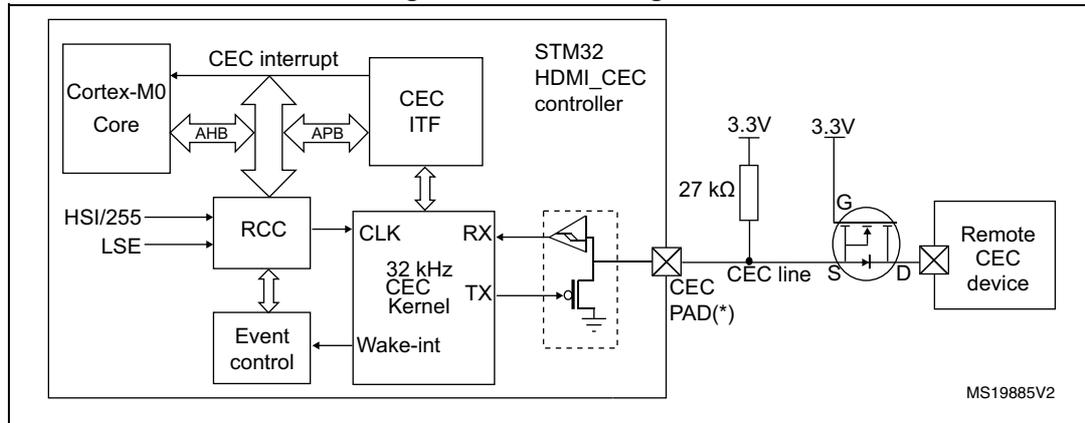
The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as Alternate Function Open Drain. The 27 kΩ pull-up must be added externally to the STM32.

To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This could be done by using a MOS transistor, as shown on [Figure 279](#).

Table 117. HDMI pin

Name	Signal type	Remarks
CEC	bidirectional	two states: 1 = high impedance 0 = low impedance A 27 kΩ must be added externally.

Figure 279. Block diagram



1. GPIO configured as output open-drain alternate function
2. When configured as output open-drain alternate function, the schmit trigger is still activated.

### 30.3.2 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 which consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.

All these blocks are made of a 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In 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 make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower which has read its own address in the header or by the follower which 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 280. Message structure

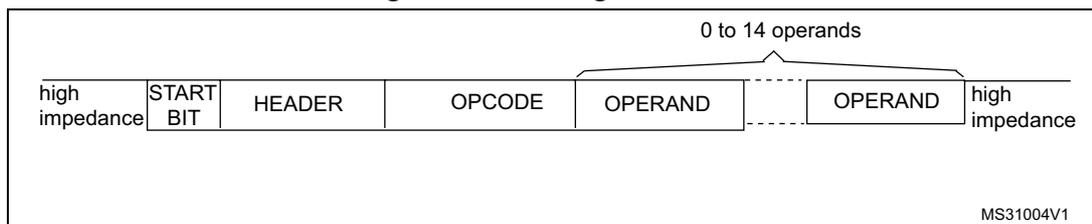
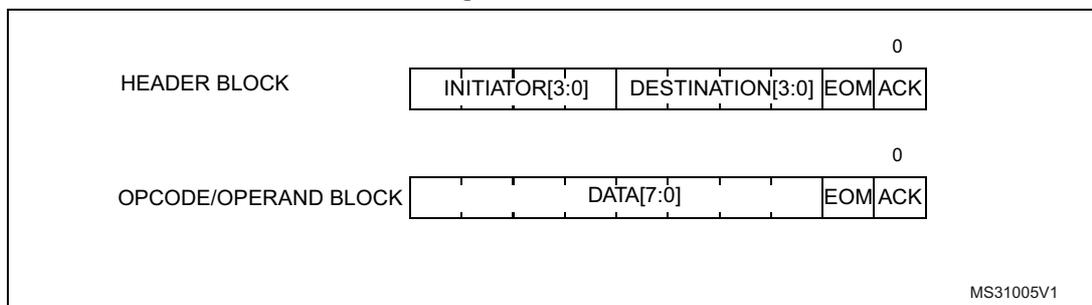


Figure 281. Blocks

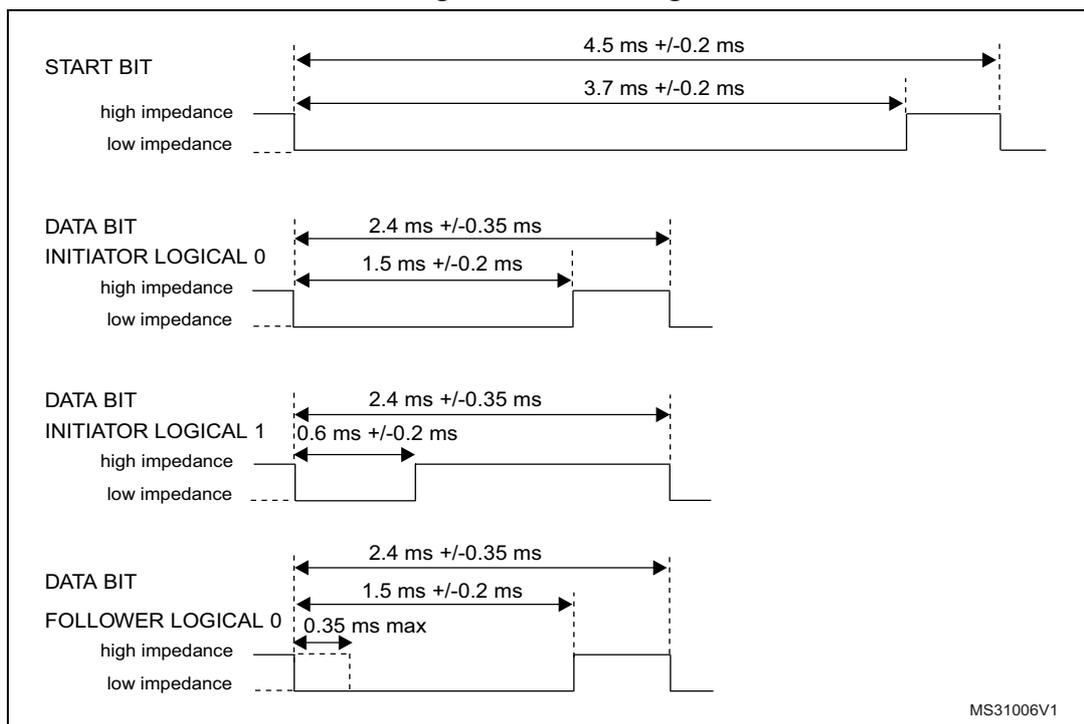


### 30.3.3 Bit timing

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.

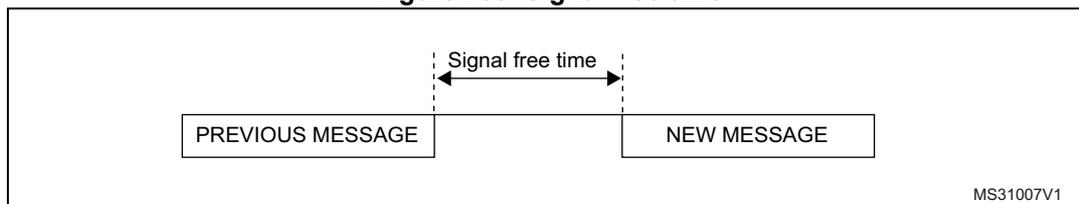
Figure 282. Bit timings



### 30.4 Arbitration

All devices that have to transmit - or retransmit - a message onto the CEC line have to ensure that it has been inactive for a number of bit periods. This signal free time is defined as the time starting from the final bit of the previous frame and depends on the initiating device and the current status as shown in the table below.

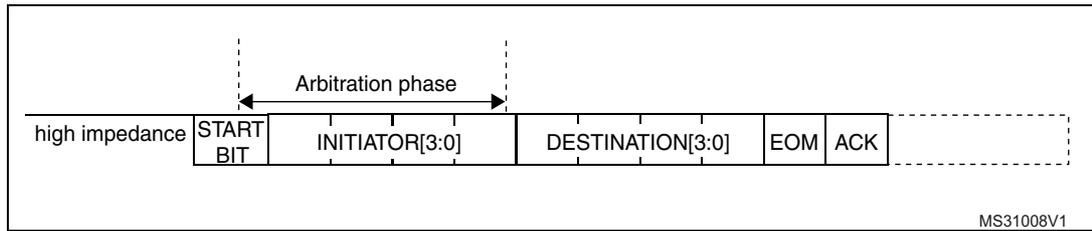
Figure 283. Signal free time



Since only one initiator is allowed at any one time, an arbitration mechanism is provided to avoid conflict when more than one initiator begins transmitting at the same time.

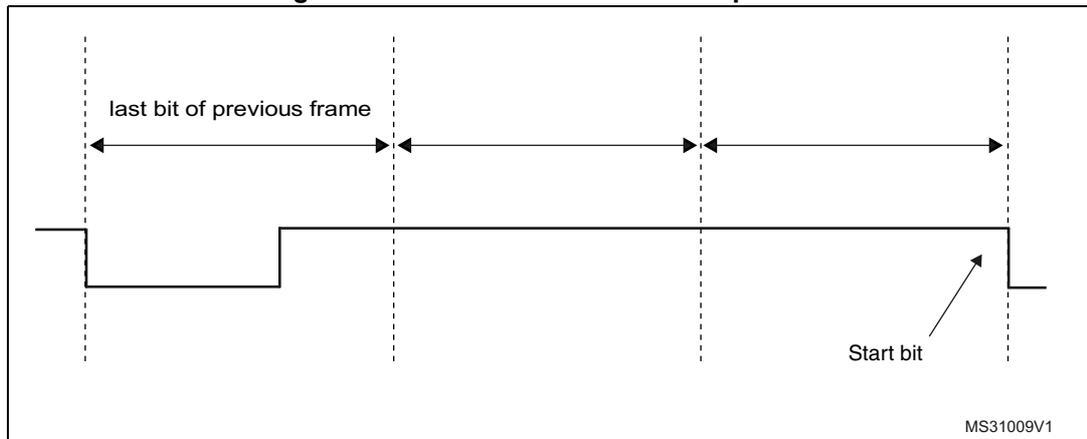
CEC line arbitration commences with the leading edge of the start bit and continues until the end of the initiator address bits within the header block. During this period, the initiator shall monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0, it then assumes it has lost arbitration, stops transmitting and becomes a follower.

Figure 284. Arbitration phase



The [Figure 285](#) shows an example for a SFT of three nominal bit periods

Figure 285. SFT of three nominal bit periods



A configurable time window is counted before starting the transmission.

In the SFT=0x0 configuration the HDMI-CEC device performs automatic SFT calculation ensuring compliancy to the HDMI-CEC Standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission

This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFTbits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

### 30.4.1 SFT option bit

In case of SFTOPT=0 configuration SFT starts being counted when the start-of-transmission command is set by software (TXSOM=1).

In case of SFTOPT=1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT

timer is still running instead, the system waits until the timer elapses before transmission can start.

In case of SFTOPT=1 a bus-event condition starting the SFT timer is detected in the following cases:

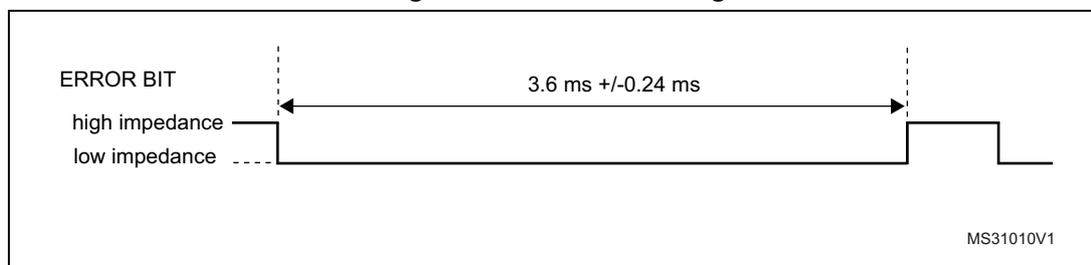
- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR=1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACK bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any un-codified low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

## 30.5 Error handling

### 30.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period, i.e. 3.6 ms nominally.

Figure 286. Error bit timing



### 30.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

### 30.5.3 Bit Rising Error (BRE)

BRE (bit rising error): is set when a bit rising edge is detected outside the windows where it is expected (see [Figure 287](#)). BRE flag also generates a CEC interrupt if the BREIE=1.

In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP=1 an error bit is generated even if BREGEN=0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN=0, BRDNOGEN=1.

### 30.5.4 Short Bit Period Error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see [Figure 287](#)). SBPE flag also generates a CEC interrupt if the SBPEIE=1.

An error bit is always generated on the line in case of a SBPE error detection. An Error Bit is not generated upon SBPE detection only when Listen mode is set (LSTN=1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN=1

### 30.5.5 Long Bit Period Error (LBPE)

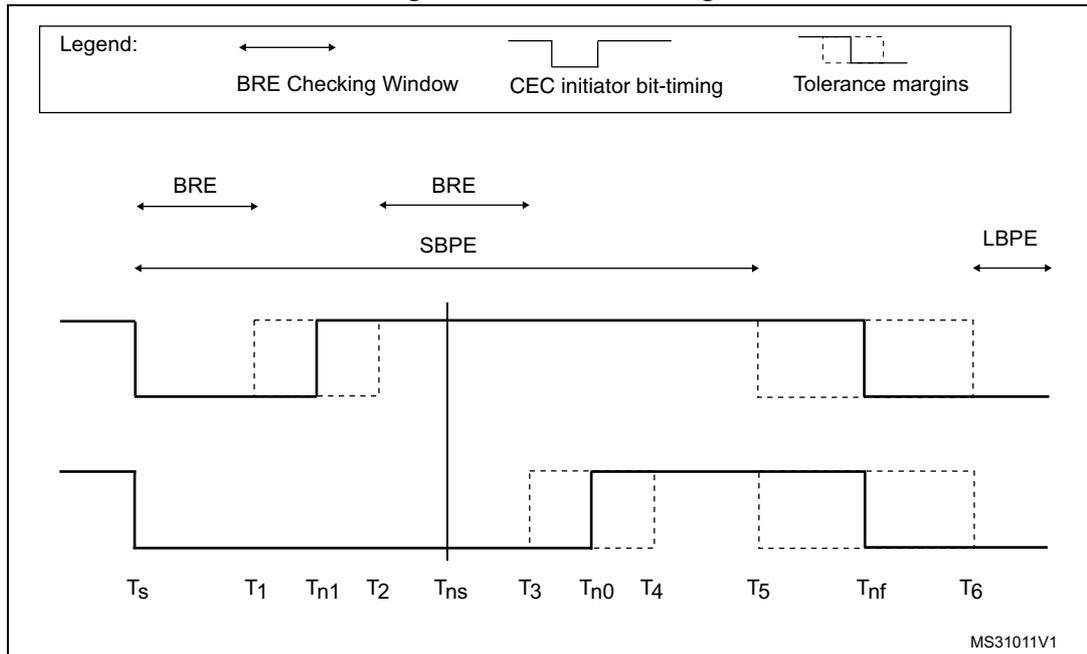
LBPE is set when a bit falling edge is not detected in a valid window (see [Figure 287](#)). LBPE flag also generates a CEC interrupt if the LBPEIE=1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN=0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN=0, BRDNOGEN=1.

*Note:* The BREGEN=1, BRESTP=0 configuration must be avoided

Figure 287. Error handling



MS31011V1

Table 118. Error handling timing parameters

Time	RXTOL	ms	Description
$T_s$	x	0	Bit start event.
$T_1$	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
$T_{n1}$	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
$T_2$	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
$T_{ns}$	x	1.05	Nominal sampling time.
$T_3$	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	
$T_{n0}$	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
$T_4$	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
$T_5$	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
$T_{nf}$	x	2.4	The nominal data bit period.
$T_6$	0	2.75	The latest time for the start of a following bit.
	1	2.95	

### 30.5.6 Transmission Error Detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE=1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to 5 times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

Figure 288. TXERR detection

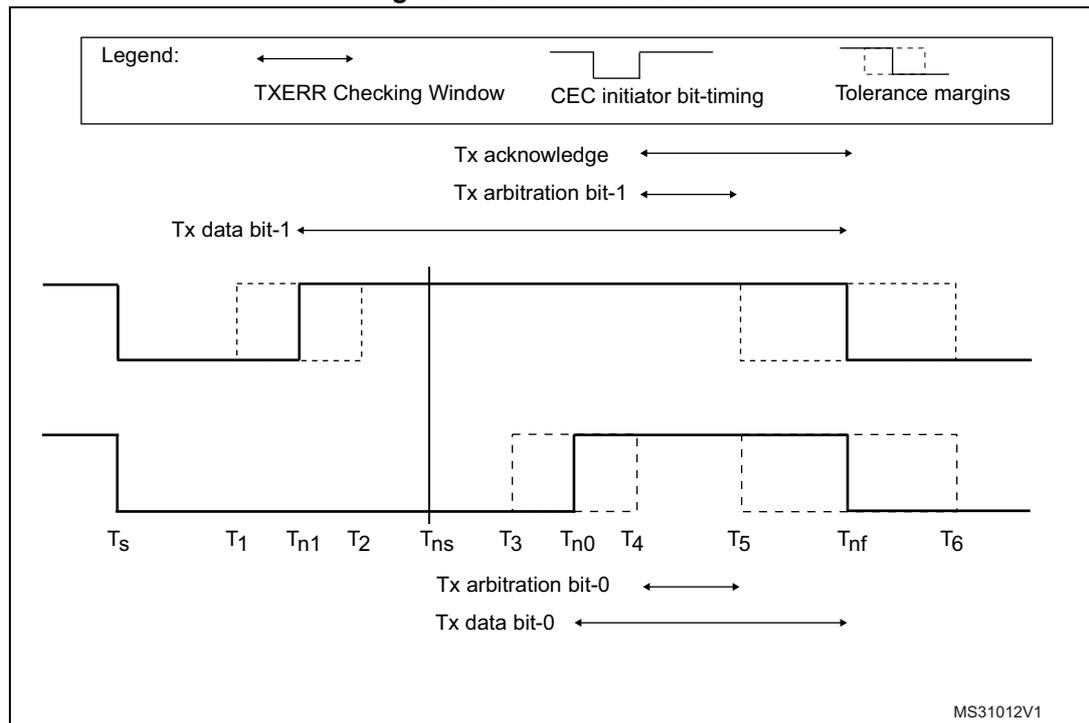


Table 119. TXERR timing parameters

Time	RXTOL	ms	Description
$T_s$	x	0	Bit start event.
$T_1$	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
$T_{n1}$	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
$T_2$	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
$T_{ns}$	x	1.05	Nominal sampling time.
$T_3$	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	

**Table 119. TXERR timing parameters (continued)**

Time	RXTOL	ms	Description
T <sub>n0</sub>	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
T <sub>4</sub>	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T <sub>5</sub>	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T <sub>nf</sub>	x	2.4	The nominal data bit period.
T <sub>6</sub>	0	2.75	The latest time for the start of a following bit.
	1	2.95	

### 30.6 HDMI-CEC interrupts

An interrupt can be produced:

- during reception if a Receive Block Transfer is finished or if a Receive Error occurs.
- during transmission if a Transmit Block Transfer is finished or if a Transmit Error occurs.

**Table 120. HDMI-CEC interrupts**

Interrupt event	Event flag	Enable Control bit
Rx-Byte Received	RXBR	RXBRIE
End of reception	RXEND	RXENDIE
Rx-Overflow	RXOVR	RXOVRIE
RxBit Rising Error	BRE	BREIE
Rx-Short Bit Period Error	SBPE	SBPEIE
Rx-Long Bit Period Error	LBPE	LBPEIE
Rx-Missing Acknowledge Error	RXACKE	RXACKEIE
Arbitration lost	ARBLST	ARBLSTIE
Tx-Byte Request	TXBR	TXBRIE
End of transmission	TXEND	TXENDIE
Tx-Buffer Underrun	TXUDR	TXUDRIE
Tx-Error	TXERR	TXERRIE
Tx-Missing Acknowledge Error	TXACKE	TXACKEIE

## 30.7 HDMI-CEC registers

Refer to [Section 1.1 on page 36](#) for a list of abbreviations used in register descriptions.

### 30.7.1 CEC control register (CEC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX EOM	TX SOM	CEC EN												
													rs	rs	rw

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

Bit 2 **TXEOM**: Tx End Of Message

The TXEOM bit is set by software to command transmission of the last byte of a CEC message.

TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.

0: TXDR data byte is transmitted with EOM=0

1: TXDR data byte is transmitted with EOM=1

*Note: TXEOM must be set when CECEN=1*

*TXEOM must be set before writing transmission data to TXDR*

*If TXEOM is set when TXSOM=0, transmitted message will consist of 1 byte (HEADER) only (PING message)*

Bit 1 **TXSOM**: Tx Start Of Message

TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM.

Start-Bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission will start after the end of reception.

TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND=1), in case of transmission underrun (TXUDR=1), negative acknowledge (TXACKE=1), and transmission error (TXERR=1). It is also cleared by CECEN=0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST=1).

TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.

- 0: No CEC transmission is on-going
- 1: CEC transmission command

Note: TXSOM must be set when CECEN=1

TXSOM must be set when transmission data is available into TXDR

HEADER's first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC\_CFGR.OAR which is used only for reception

Bit 0 **CECEN**: CEC Enable

The CECEN bit is set and cleared by software. CECEN=1 starts message reception and enables the TXSOM control. CECEN=0 disables the CEC peripheral, clears all bits of CEC\_CR register and aborts any on-going reception or transmission.

- 0: CEC peripheral is off
- 1: CEC peripheral is on

### 30.7.2 CEC configuration register (CEC\_CFGR)

This register is used to configure the HDMI-CEC controller.

Address offset: 0x04

Reset value: 0x0000 0000

**Caution:** It is mandatory to write CEC\_CFGR only when CECEN=0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSTN	OAR[14:0]														
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFT OPT	BRDN OGEN	LBPE GEN	BRE GEN	BRE STP	RX TOL	SFT[2:0]		
							rw	rw	rw	rw	rw	rw	rw		

Bit 31 **LSTN**: Listen mode

LSTN bit is set and cleared by software.

0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.

1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

Bits 30:16 **OAR**: Own addresses configuration

The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.

At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN=1), but without acknowledge sent. Broadcast messages are always received.

Example:

OAR = 0b000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5.

Consequently, each message directed to one of these addresses is received.

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

Bit 8 **SFTOP**: *SFT Option Bit*

The SFTOPT bit is set and cleared by software.

0: SFT timer starts when TXSOM is set by software

1: SFT timer starts automatically at the end of message transmission/reception.

Bit 7 **BRDNOGEN**: *Avoid Error-Bit Generation in Broadcast*

The BRDNOGEN bit is set and cleared by software.

0: BRE detection with BRESTP=1 and BREGEN=0 on a broadcast message generates an Error-Bit on the CEC line. LBPE detection with LBPEGEN=0 on a broadcast message generates an Error-Bit on the CEC line

1: Error-Bit is not generated in the same condition as above. An Error-Bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.

Bit 6 **LBPEGEN**: *Generate Error-Bit on Long Bit Period Error*

The LBPEGEN bit is set and cleared by software.

0: LBPE detection does not generate an Error-Bit on the CEC line

1: LBPE detection generates an Error-Bit on the CEC line

*Note: If BRDNOGEN=0, an Error-bit is generated upon LBPE detection in broadcast even if LBPEGEN=0*

Bit 5 **BREGEN**: *Generate Error-Bit on Bit Rising Error*

The BREGEN bit is set and cleared by software.

0: BRE detection does not generate an Error-Bit on the CEC line

1: BRE detection generates an Error-Bit on the CEC line (if BRESTP is set)

*Note: If BRDNOGEN=0, an Error-bit is generated upon BRE detection with BRESTP=1 in broadcast even if BREGEN=0*

**Bit 4 BRESTP: Rx-Stop on Bit Rising Error**

The BRESTP bit is set and cleared by software.

- 0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.
- 1: BRE detection stops message reception

**Bit 3 RXTOL: Rx-Tolerance**

The RXTOL bit is set and cleared by software.

- 0: Standard tolerance margin:
  - Start-Bit, +/- 200  $\mu$ s rise, +/- 200  $\mu$ s fall.
  - Data-Bit: +/- 200  $\mu$ s rise. +/- 350  $\mu$ s fall.
- 1: Extended Tolerance
  - Start-Bit: +/- 400  $\mu$ s rise, +/- 400  $\mu$ s fall
  - Data-Bit: +/-300  $\mu$ s rise, +/- 500  $\mu$ s fall

**Bits 2:0 SFT: Signal Free Time**

SFT bits are set by software. In the SFT=0x0 configuration the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.

- " 0x0
  - 2.5 Data-Bit periods if CEC is the last bus initiator with unsuccessful transmission (ARBLST=1, TXERR=1, TXUDR=1 or TXACKE= 1)
  - 4 Data-Bit periods if CEC is the new bus initiator
  - 6 Data-Bit periods if CEC is the last bus initiator with successful transmission (TXEOM=1)
- " 0x1: 0.5 nominal data bit periods
- " 0x2: 1.5 nominal data bit periods
- " 0x3: 2.5 nominal data bit periods
- " 0x4: 3.5 nominal data bit periods
- " 0x5: 4.5 nominal data bit periods
- " 0x6: 5.5 nominal data bit periods
- " 0x7: 6.5 nominal data bit periods

### 30.7.3 CEC Tx data register (CEC\_TXDR)

Address offset: 0x8

Reset value: 0x0000 0000

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

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

Bits 7:0 **TXD[7:0]**: Tx Data register.

TXD is a write-only register containing the data byte to be transmitted.

Note: TXD must be written when TXSTART=1

### 30.7.4 CEC Rx Data Register (CEC\_RXDR)

Address offset: 0xC

Reset value: 0x0000 0000

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

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

Bits 7:0 **RXD[7:0]**: Rx Data register.

RXD is read-only and contains the last data byte which has been received from the CEC line.

### 30.7.5 CEC Interrupt and Status Register (CEC\_ISR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TX ACKE	TX ERR	TX UDR	TX END	TXBR	ARB LST	RX ACKE	LBPE	SBPE	BRE	RX OVR	RX END	RXBR
			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 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKE**: Tx-Missing Acknowledge Error

In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.

TXACKE is cleared by software write at 1.

Bit 11 **TXERR**: Tx-Error

In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.

TXERR is cleared by software write at 1.

Bit 10 **TXUDR**: Tx-Buffer Underrun

In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.

TXUDR is cleared by software write at 1

Bit 9 **TXEND**: End of Transmission

TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.

TXEND is cleared by software write at 1.

Bit 8 **TXBR**: Tx-Byte Request

TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within 6 nominal data-bit periods before transmission underrun error occurs (TXUDR).

TXBR is cleared by software write at 1.

Bit 7 **ARBLST**: Arbitration Lost

ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.

ARBLST is cleared by software write at 1.

Bit 6 **RXACKE**: Rx-Missing Acknowledge

In receive mode, RXACKE is set by hardware to inform application that no acknowledge was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception. RXACKE is cleared by software write at 1.

Bit 5 **LBPE**: Rx-Long Bit Period Error

LBPE is set by hardware in case a Data-Bit waveform is detected with Long Bit Period Error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an Error-Bit on the CEC line if LBPEGEN=1. In case of broadcast, Error-Bit is generated even in case of LBPEGEN=0.

LBPE is cleared by software write at 1.

Bit 4 **SBPE**: Rx-Short Bit Period Error

SBPE is set by hardware in case a Data-Bit waveform is detected with Short Bit Period Error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an Error-Bit on the CEC line.

SBPE is cleared by software write at 1.

Bit 3 **BRE**: Rx-Bit Rising Error

BRE is set by hardware in case a Data-Bit waveform is detected with Bit Rising Error. BRE is set either at the time the misplaced rising edge occurs, or at the end of the maximum BRE tolerance allowed by RXTOL, in case rising edge is still longing. BRE stops message reception if BRESTP=1. BRE generates an Error-Bit on the CEC line if BREGEN=1.  
BRE is cleared by software write at 1.

Bit 2 **RXOVR**: Rx-Overrun

RXOVR is set by hardware if RXBR is not yet cleared at the time a new byte is received on the CEC line and stored into RXD. RXOVR assertion stops message reception so that no acknowledge is sent. In case of broadcast, a negative acknowledge is sent.  
RXOVR is cleared by software write at 1.

Bit 1 **RXEND**: End Of Reception

RXEND is set by hardware to inform application that the last byte of a CEC message is received from the CEC line and stored into the RXD buffer. RXEND is set at the same time of RXBR.  
RXEND is cleared by software write at 1.

Bit 0 **RXBR**: Rx-Byte Received

The RXBR bit is set by hardware to inform application that a new byte has been received from the CEC line and stored into the RXD buffer.  
RXBR is cleared by software write at 1.

### 30.7.6 CEC interrupt enable register (CEC\_IER)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACK IE	TXERR IE	TX UDRIE	TXEND IE	TXBR IE	ARBLST IE	RXACK IE	LBPE IE	SBPE IE	BREIE	RXOVR IE	RXEND IE	RXBR IE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKIE**: Tx-Missing Acknowledge Error Interrupt Enable

The TXACKIE bit is set and cleared by software.  
0: TXACKIE interrupt disabled  
1: TXACKIE interrupt enabled

Bit 11 **TXERRIE**: Tx-Error Interrupt Enable

The TXERRIE bit is set and cleared by software.  
0: TXERR interrupt disabled  
1: TXERR interrupt enabled

Bit 10 **TXUDRIE**: Tx-Underrun Interrupt Enable

The TXUDRIE bit is set and cleared by software.  
0: TXUDR interrupt disabled  
1: TXUDR interrupt enabled

- Bit 9 **TXENDIE**: Tx-End Of Message Interrupt Enable  
The TXENDIE bit is set and cleared by software.  
0: TXEND interrupt disabled  
1: TXEND interrupt enabled
- Bit 8 **TXBRIE**: Tx-Byte Request Interrupt Enable  
The TXBRIE bit is set and cleared by software.  
0: TXBR interrupt disabled  
1: TXBR interrupt enabled
- Bit 7 **ARBLSTIE**: Arbitration Lost Interrupt Enable  
The ARBLSTIE bit is set and cleared by software.  
0: ARBLST interrupt disabled  
1: ARBLST interrupt enabled
- Bit 6 **RXACKIE**: Rx-Missing Acknowledge Error Interrupt Enable  
The RXACKIE bit is set and cleared by software.  
0: RXACKE interrupt disabled  
1: RXACKE interrupt enabled
- Bit 5 **LBPEIE**: Long Bit Period Error Interrupt Enable  
The LBPEIE bit is set and cleared by software.  
0: LBPE interrupt disabled  
1: LBPE interrupt enabled
- Bit 4 **SBPEIE**: Short Bit Period Error Interrupt Enable  
The SBPEIE bit is set and cleared by software.  
0: SBPE interrupt disabled  
1: SBPE interrupt enabled
- Bit 3 **BREIE**: Bit Rising Error Interrupt Enable  
The BREIE bit is set and cleared by software.  
0: BRE interrupt disabled  
1: BRE interrupt enabled
- Bit 2 **RXOVRIE**: Rx-Buffer Overrun Interrupt Enable  
The RXOVRIE bit is set and cleared by software.  
0: RXOVR interrupt disabled  
1: RXOVR interrupt enabled
- Bit 1 **RXENDIE**: End Of Reception Interrupt Enable  
The RXENDIE bit is set and cleared by software.  
0: RXEND interrupt disabled  
1: RXEND interrupt enabled
- Bit 0 **RXBRIE**: Rx-Byte Received Interrupt Enable  
The RXBRIE bit is set and cleared by software.  
0: RXBR interrupt disabled  
1: RXBR interrupt enabled

**Caution:** (\*) It is mandatory to write CEC\_IER only when CECEN=0

### 30.7.7 HDMI-CEC register map

The following table summarizes the HDMI-CEC registers.

**Table 121. 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_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																
0x04	CEC_CFGR	LSTN	OAR[14:0]															SFTOPT	BRDNOGEN	LBPEGEN	BREGEN	BRESTP	RXTOL	SFT[2:0]									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x08	CEC_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C	CEC_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]								
	Reset value																																
0x10	CEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSLTI	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	CEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSLTI	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2.2 on page 39](#) for the register boundary addresses.

# 31 Debug support (DBG)

This section applies to the whole STM32F37xxx family, unless otherwise specified.

## 31.1 Overview

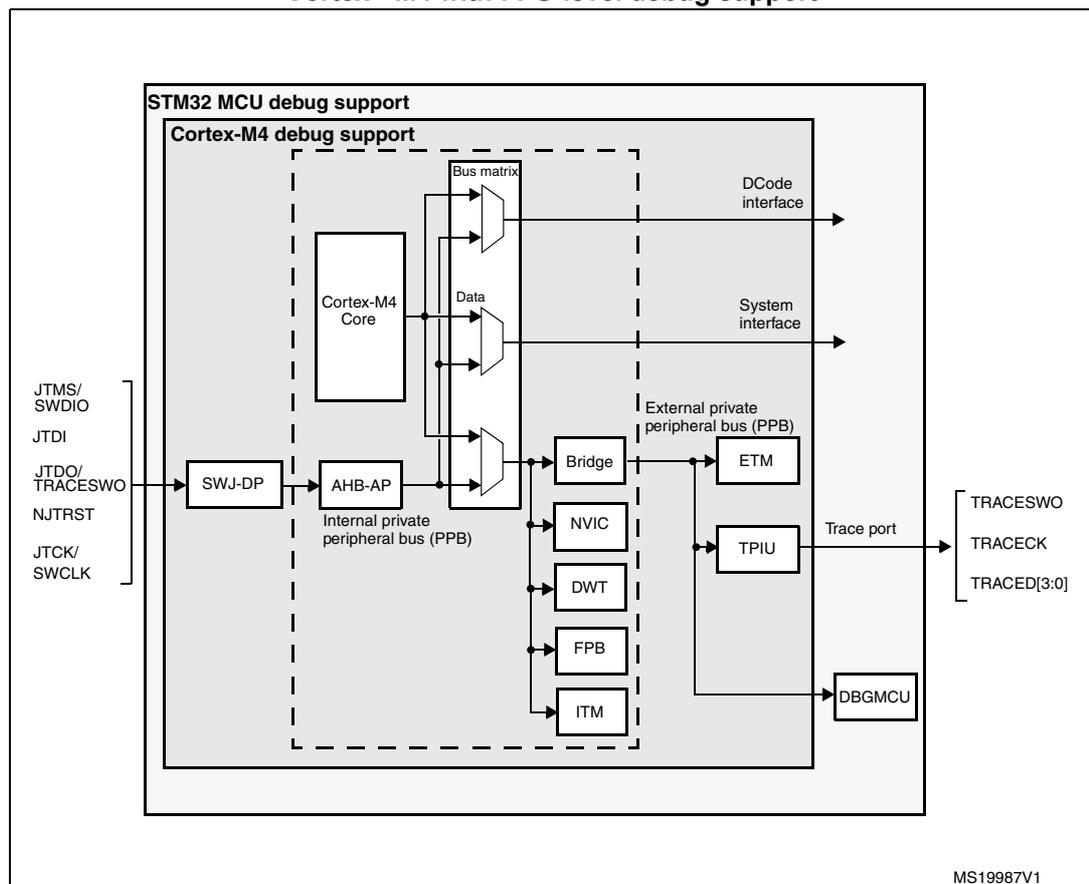
The STM32F37xxx built around a Cortex<sup>®</sup>-M4 with FPU core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state 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 STM32F37xxx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

**Figure 289. Block diagram of STM32F37xx MCU and Cortex<sup>®</sup>-M4 with FPU-level debug support**



1. The debug features embedded in the Cortex<sup>®</sup>-M4 with FPU core are a subset of the ARM CoreSight

Design Kit.

The ARM Cortex<sup>®</sup>-M4 with FPU 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
- TPUJ: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)
- ETM: Embedded Trace Macrocell (available on larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32F37xxx:

- 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<sup>®</sup>-M4 with FPU core, refer to the Cortex<sup>®</sup>-M4 with FPU Technical Reference Manual and to the CoreSight Design Kit-r1p0 TRM (see [Section 31.2: Reference ARM documentation](#)).

## 31.2 Reference ARM documentation

- Cortex<sup>®</sup>-M4 with FPU Technical Reference Manual (TRM)  
It is available from:  
<http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439c/index.html>
- ARM Debug Interface V5
- ARM CoreSight Design Kit revision r1p1 Technical Reference Manual

## 31.3 SWJ debug port (serial wire and JTAG)

The STM32F37xxx 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 290. SWJ debug port

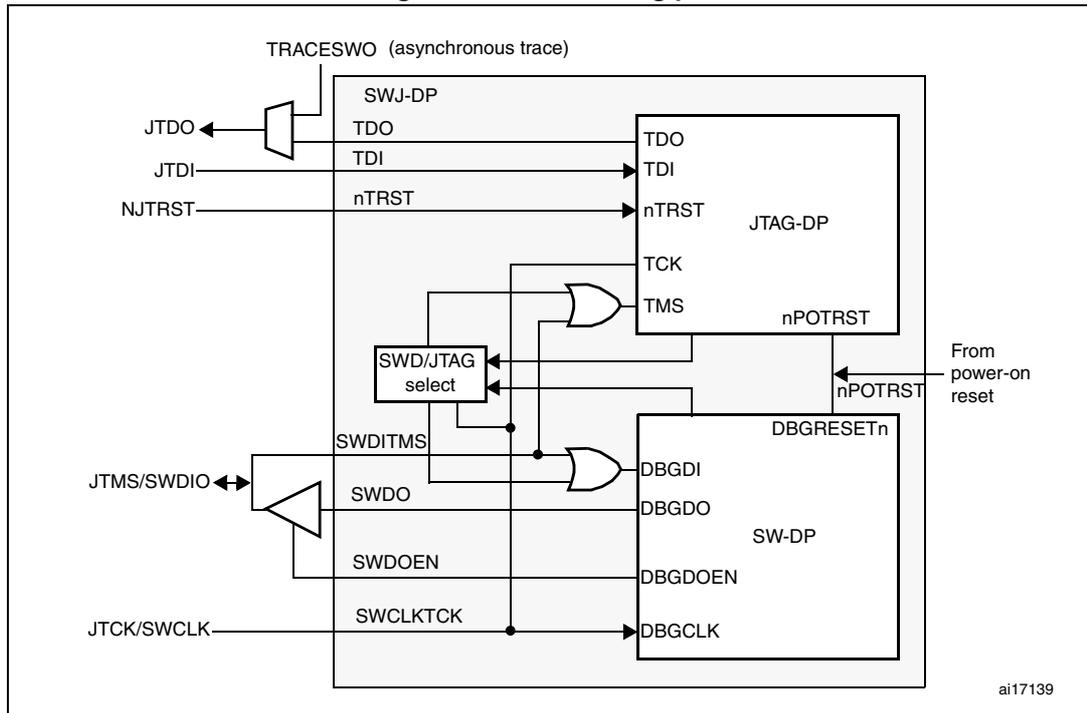


Figure 290 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

### 31.3.1 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

## 31.4 Pinout and debug port pins

The STM32F37xxx 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.

### 31.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F37xxx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

**Table 122. SWJ debug port pins**

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assignment
	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 asynchronous trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

### 31.4.2 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 STM32F37xxx MCU offer the possibility of disabling some or all of the SWJ-DP ports and so, of releasing the associated pins for general-purpose I/O

(GPIO) usage. For more details on how to disable SWJ-DP port pins, please refer to [Section 8.3.2: I/O pin alternate function multiplexer and mapping](#).

**Table 123. Flexible SWJ-DP pin assignment**

Available debug ports	SWJ I/O 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 I/O pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

### 31.4.3 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 I/O 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 I/O 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.*

### 31.4.4 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 change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO\_MODER register. 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 tri-state (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 I/O pin configuration in the IOPORT controller has no effect.*

## 31.5 STM32F37xxx JTAG TAP connection

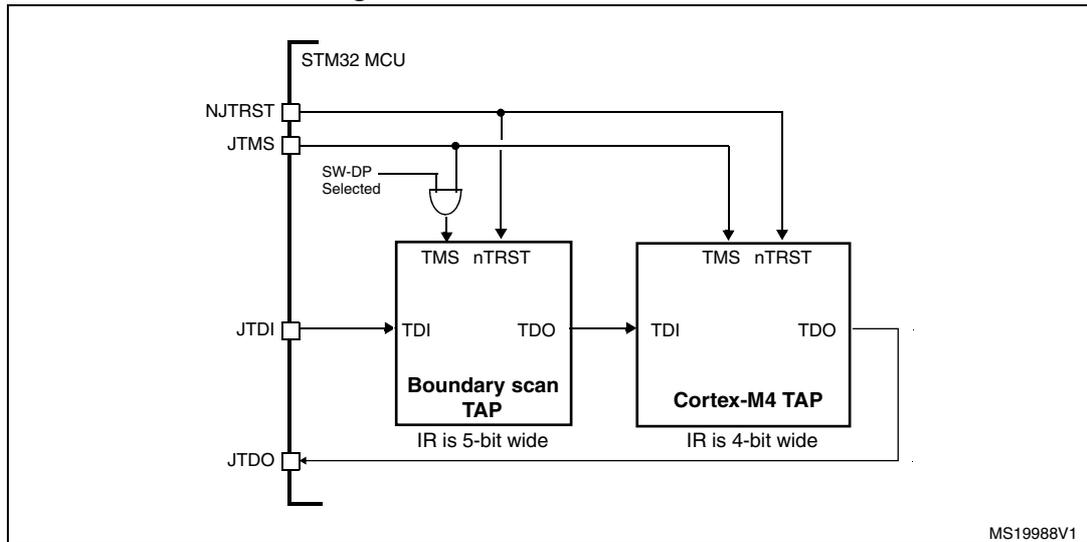
The STM32F37xxx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex<sup>®</sup>-M4 with FPU TAP (IR is 4-bit wide).

To access the TAP of the Cortex<sup>®</sup>-M4 with FPU 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 291. JTAG TAP connections



### 31.6 ID codes and locking mechanism

There are several ID codes inside the STM32F37xxx 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.

### 31.6.1 MCU device ID code

The STM32F37xxx 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 31.16 on page 874](#)). 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: 0xE004 2000

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
Res.				DEV_ID											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

This code is read as 0xE004 2000 for Revision 1.0

- Bits 31:16 **REV\_ID(15:0)** Revision identifier  
 This field indicates the revision of the device.  
 0x1000: Revision A  
 0x2000 for revision B  
 ...
- Bits 15:12 Reserved, must be kept at reset value.
- Bits 11:0 **DEV\_ID(11:0)**: Device identifier  
 This field indicates the device and its revision.  
 The device ID is 0x432.

### 31.6.2 Boundary scan TAP

#### JTAG ID code

The TAP of the STM32F37xxx BSC (boundary scan) integrates a JTAG ID code equal to 0x06432041 0x4BA00477.

### 31.6.3 Cortex<sup>®</sup>-M4 with FPU TAP

The TAP of the ARM Cortex<sup>®</sup>-M4 with FPU 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 **0x4BA00477** (corresponds to Cortex<sup>®</sup>-M4 with FPU, see [Section 31.2: Reference ARM documentation](#)).

Only the DEV\_ID(11:0) should be used for identification by the debugger/programmer tools.

### 31.6.4 Cortex<sup>®</sup>-M4 with FPU JEDEC-106 ID code

The ARM Cortex<sup>®</sup>-M4 with FPU 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.

## 31.7 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<sup>®</sup>-M4 with FPU *Technical Reference Manual (TRM)*, for references, please see [Section 31.2: Reference ARM documentation](#)).

**Table 124. JTAG debug port data registers**

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	ID CODE 0x3BA00477 (ARM Cortex <sup>®</sup> -M4 with FPU 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 <a href="#">Table 125</a> for a description of the A(3:2) bits

**Table 124. JTAG debug port data registers (continued)**

IR(3:0)	Data register	Details
1011	APACC [35 bits]	<p>Access port access register Initiates an access port and allows access to an access port register.</p> <ul style="list-style-type: none"> <li>– When transferring data IN:                             <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request</li> <li>Bits 2:1 = A[3:2] = 2-bit address (sub-address AP registers).</li> <li>Bit 0 = RnW= Read request (1) or write request (0).</li> </ul> </li> <li>– When transferring data OUT:                             <ul style="list-style-type: none"> <li>Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request</li> <li>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:                                     <ul style="list-style-type: none"> <li>010 = OK/FAULT</li> <li>001 = WAIT</li> <li>OTHER = reserved</li> </ul> </li> </ul> </li> </ul> <p>There are many AP Registers (see AHB-AP) addressed as the combination of:</p> <ul style="list-style-type: none"> <li>– The shifted value A[3:2]</li> <li>– The current value of the DP SELECT register</li> </ul>
1000	ABORT [35 bits]	<p>Abort register</p> <ul style="list-style-type: none"> <li>– Bits 31:1 = Reserved</li> <li>– Bit 0 = DAPABORT: write 1 to generate a DAP abort.</li> </ul>

**Table 125. 32-bit debug port registers addressed through the shifted value A[3:2]**

Address	A(3:2) value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	<p>DP CTRL/STAT register. Used to:</p> <ul style="list-style-type: none"> <li>– Request a system or debug power-up</li> <li>– Configure the transfer operation for AP accesses</li> <li>– Control the pushed compare and pushed verify operations.</li> <li>– Read some status flags (overrun, power-up acknowledges)</li> </ul>
0x8	10	<p>DP SELECT register: Used to select the current access port and the active 4-words register window.</p> <ul style="list-style-type: none"> <li>– Bits 31:24: APSEL: select the current AP</li> <li>– Bits 23:8: reserved</li> <li>– Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP</li> <li>– Bits 3:0: reserved</li> </ul>
0xC	11	<p>DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)</p>

## 31.8 SW debug port

### 31.8.1 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.

### 31.8.2 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 126. 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 <a href="#">Table 125</a> )
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<sup>®</sup>-M4 with FPU *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 127. 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 128. 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.

### 31.8.3 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<sup>®</sup>-M4 with FPU).

*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<sup>®</sup>-M4 with FPU TRM and the CoreSight Design Kit r1p0 TRM.

### 31.8.4 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.

### 31.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 129. 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 <b>0x2BA01477</b> (identifies the SW-DP)
00	Write		ABORT	
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: <ul style="list-style-type: none"> <li>– request a system or debug power-up</li> <li>– configure the transfer operation for AP accesses</li> <li>– control the pushed compare and pushed verify operations.</li> <li>– read some status flags (overrun, power-up acknowledges)</li> </ul>
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

### 31.8.6 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

### 31.9 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 registers 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<sup>®</sup>-M4 with FPU includes 9 x 32-bits registers:

**Table 130. Cortex<sup>®</sup>-M4 with FPU 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<sup>®</sup>-M4 with FPU TRM for further details.

### 31.10 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 131. 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 <b>TRCENA</b> which enable the use of a TRACE.

These registers are not reset by a system reset. They are only reset by a power-on reset. Refer to the Cortex<sup>®</sup>-M4 with FPU 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.

### 31.11 Capability of the debugger host to connect under system reset

The STM32F37xxx 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<sup>®</sup>-M4 with FPU 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.*

## 31.12 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.

## 31.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM 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

## 31.14 ITM (instrumentation trace macrocell)

### 31.14.1 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<sup>®</sup>-M4 with FPU 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.

### 31.14.2 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 132. 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		

Table 132. Main ITM registers

Address	Register	Details
@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 31.17.2: TRACE pin assignment](#) and [Section 31.16.3: 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)

## 31.15 ETM (Embedded trace macrocell)

### 31.15.1 General description

The ETM enables the reconstruction of program execution. Data are traced using the Data Watchpoint and Trace (DWT) component or the Instruction Trace Macrocell (ITM) whereas instructions are traced using the Embedded Trace Macrocell (ETM).

The ETM transmits information as packets and is triggered by embedded resources. These resources must be programmed independently and the trigger source is selected using the Trigger Event Register (0xE0041008). An event could be a simple event (address match from an address comparator) or a logic equation between 2 events. The trigger source is one of the fourth comparators of the DWT module, The following events can be monitored:

- Clock cycle matching
- Data address matching

For more informations on the trigger resources refer to [Section 31.13: DWT \(data watchpoint trigger\)](#).

The packets transmitted by the ETM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to [Section 31.17: TPIU \(trace port](#)

*interface unit*) and then outputs the complete packet sequence to the debugger host.

### 31.15.2 Signal protocol, packet types

This part is described in the chapter 7 ETMv3 Signal Protocol of the ARM IHI 0014N document.

### 31.15.3 Main ETM registers

For more information on registers refer to the chapter 3 of the ARM IHI 0014N specification.

**Table 133. Main ETM registers**

Address	Register	Details
0xE0041FB0	ETM Lock Access	Write 0xC5ACCE55 to unlock the write access to the other ETM registers.
0xE0041000	ETM Control	This register controls the general operation of the ETM, for instance how tracing is enabled.
0xE0041010	ETM Status	This register provides information about the current status of the trace and trigger logic.
0xE0041008	ETM Trigger Event	This register defines the event that will control trigger.
0xE004101C	ETM Trace Enable Control	This register defines which comparator is selected.
0xE0041020	ETM Trace Enable Event	This register defines the trace enabling event.
0xE0041024	ETM Trace Start/Stop	This register defines the traces used by the trigger source to start and stop the trace, respectively.

### 31.15.4 Configuration example

To output a simple value to the TPIU:

- Configure the TPIU and enable the I/O\_TRACEN to assign TRACE I/Os in the STM32F37xxx debug configuration register.
- Write 0xC5ACCE55 to the ETM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00001D1E to the control register (configure the trace)
- Write 0000406F to the Trigger Event register (define the trigger event)
- Write 0000006F to the Trace Enable Event register (define an event to start/stop)
- Write 00000001 to the Trace Start/stop register (enable the trace)
- Write 0000191E to the ETM Control Register (end of configuration)

## 31.16 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I2C and bxCAN during a breakpoint
- Control of the trace pins assignment

### 31.16.1 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.

*Note:* When the device is in Debug Stop or Debug Sleep mode (configured through DBG\_STOP or DBG\_SLEEP bits in the DBGMCU\_CR registers), the SysTick timer is always running and cannot be frozen. As a consequence, different SysTick timer values are observed if the application code is running with and without debugger.

### 31.16.2 Debug support for timers, watchdog, bxCAN and I<sup>2</sup>C

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 bxCAN, the user can choose to block the update of the receive register during a breakpoint.

For the I<sup>2</sup>C, the user can choose to block the SMBUS timeout during a breakpoint.

### 31.16.3 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
- bxCAN communication 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TRACE_MODE [1:0]		TRACE_IOEN	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP							
								rw	rw	rw			rw	rw	rw

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

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).

### 31.16.4 Debug MCU APB1 freeze register (DBGMCU\_APB1\_FZ)

The DBGMCU\_APB1\_FZ register is used to configure the MCU under DEBUG. It concerns the APB1 peripherals:

- Timer clock counter freeze
- I2C SMBUS timeout freeze
- Window watchdog and independent watchdog counter freeze support

This DBGMCU\_APB1\_FZ is mapped on the external PPB bus at address 0xE0042008.

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 2008

Only 32-bit access are supported.

Power on reset (POR): 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	DBG_CAN_STOP	Res.	Res.	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Res.	Res.	Res.	Res.	Res.
						rw			rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	DBG_TIM18_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **DBG\_CAN\_STOP**: CAN stopped when core is halted

- 0: Same behavior as in normal mode
- 1: The CAN receive registers are frozen

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **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 21 **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 20:13 Reserved, must be kept at reset value.

- Bit 12 **DBG\_IWDG\_STOP**: Independent watchdog stopped when core is halted  
0: The independent watchdog counter clock continues even if the core is halted  
1: The independent watchdog counter clock is stopped when the core is halted
- Bit 11 **DBG\_WWDG\_STOP**: 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 10 **DBG\_RTC\_STOP**: RTC stopped when core is halted  
0: The clock of the RTC counter is fed even if the core is halted  
1: The clock of the RTC counter is stopped when the core is halted  
*Note: This bit is available only in high density devices.*
- Bit 9 **DBG\_TIM18\_STOP**: TIM18 counter stopped when core is halted  
0: The counter clock of TIM18 is fed even if the core is halted  
1: The counter clock of TIM18 is stopped when the core is halted
- Bit 8 **DBG\_TIM14\_STOP**: TIM14 counter stopped when core is halted  
0: The counter clock of TIM14 is fed even if the core is halted  
1: The counter clock of TIM14 is stopped when the core is halted
- Bit 7 **DBG\_TIM13\_STOP**: TIM13 counter stopped when core is halted  
0: The counter clock of TIM13 is fed even if the core is halted  
1: The counter clock of TIM13 is stopped when the core is halted
- Bit 6 **DBG\_TIM12\_STOP**: TIM12 counter stopped when core is halted  
0: The counter clock of TIM12 is fed even if the core is halted  
1: The counter clock of TIM12 is stopped when the core is halted
- Bit 5 **DBG\_TIM7\_STOP**: TIM7 counter stopped when core is halted  
0: The counter clock of TIM7 is fed even if the core is halted  
1: The counter clock of TIM7 is stopped when the core is halted
- Bit 4 **DBG\_TIM6\_STOP**: TIM6 counter stopped when core is halted  
0: The counter clock of TIM6 is fed even if the core is halted  
1: The counter clock of TIM6 is stopped when the core is halted
- Bit 3 **DBG\_TIM5\_STOP**: TIM5 counter stopped when core is halted  
0: The counter clock of TIM5 is fed even if the core is halted  
1: The counter clock of TIM5 is stopped when the core is halted
- Bit 2 **DBG\_TIM4\_STOP**: TIM4 counter stopped when core is halted  
0: The counter clock of TIM4 is fed even if the core is halted  
1: The counter clock of TIM4 is stopped when the core is halted
- Bit 1 **DBG\_TIM3\_STOP**: TIM3 counter stopped when core is halted  
0: The counter clock of TIM3 is fed even if the core is halted  
1: The counter clock of TIM3 is stopped when the core is halted
- Bit 0 **DBG\_TIM2\_STOP**: TIM2 counter stopped when core is halted  
0: The counter clock of TIM2 is fed even if the core is halted  
1: The counter clock of TIM2 is stopped when the core is halted

### 31.16.5 Debug MCU APB2 freeze register (DBGMCU\_APB2\_FZ)

The DBGMCU\_APB2\_FZ register is used to configure the MCU under DEBUG. It concerns APB2 peripherals:

- Timer clock counter freeze

This register is mapped on the external PPB bus at address 0xE004 200C

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 200C

Only 32-bit access is supported.

POR: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DBG_TIM19_STOP	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.									
										rw	rw	rw	rw		

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

Bits 5:2 **DBG\_TIMx\_STOP**: TIMx counter stopped when core is halted (x=15..19)

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 1:9 Reserved, must be kept at reset value.

## 31.17 TPIU (trace port interface unit)

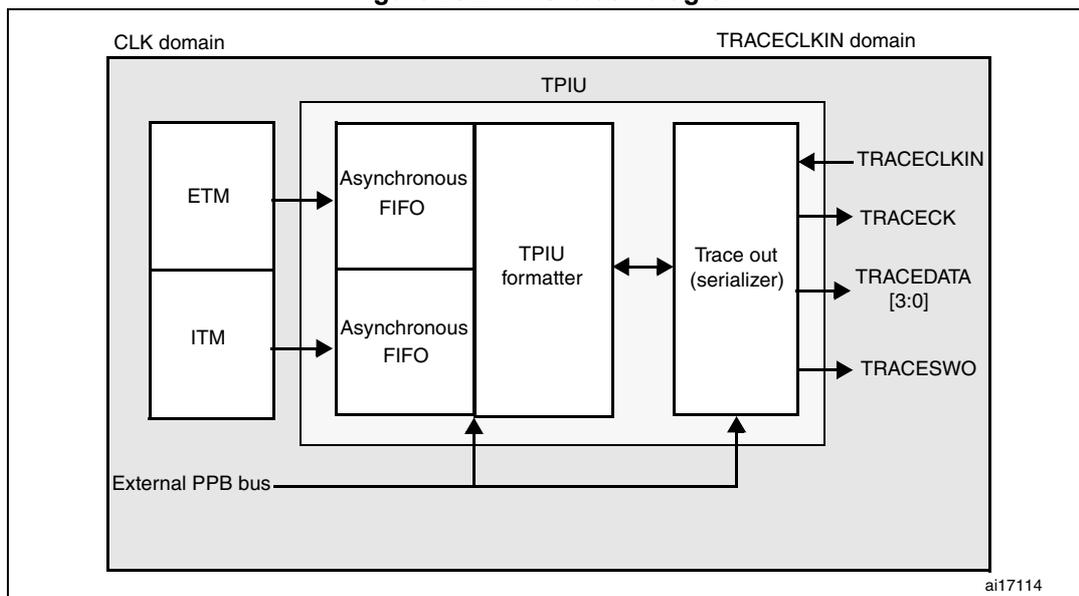
### 31.17.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

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 292. TPIU block diagram



### 31.17.2 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 134. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F37xxx 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 135. Synchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F37xxx 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 136. Flexible TRACE pin assignment**

DBGMCU_CR register		Pins assigned for:	TRACE I/O 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 <sup>(1)</sup>					
1	00	Asynchronous Trace	TRACESWO				Released (usable as GPIO)	
1	01	Synchronous Trace 1 bit	Released <sup>(1)</sup>	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

### 31.17.3 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

### 31.17.4 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.

### 31.17.5 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 TPIU 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 TPIU (trace source ID added).

### 31.17.6 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.

### 31.17.7 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 STM32F37xxx 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%.

### 31.17.8 TRACECLKIN connection inside the STM32F37xxx

In the STM32F37xxx, 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 STM32F37xxx 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.*

### 31.17.9 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 137. 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 = TriglIn = 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 <b>Note:</b> 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 <sup>®</sup> -M4 with FPU, always read as 0x00000008

### 31.17.10 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

## 31.18 DBG register map

The following table summarizes the Debug registers

**Table 138. 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														Res.	Res.	Res.	Res.	DEV_ID													
	Reset value <sup>(1)</sup>	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																									0	0	0					
0xE004 2008	DBGMCU_APB1_FZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value							0			0	0										0	0				0	0		0	0	0	0



Table 138. DBG register map and reset values (continued)

Addr.	Register	
0xE004 200C	DBGMCU_APB2_FZ	31
		Res.
		30
		Res.
		29
		Res.
		28
		Res.
		27
		Res.
		26
		Res.
		25
		Res.
		24
		Res.
		23
		Res.
		22
		Res.
21		
Res.		
20		
Res.		
19		
Res.		
18		
Res.		
17		
Res.		
16		
Res.		
15		
Res.		
14		
Res.		
13		
Res.		
12		
Res.		
11		
Res.		
10		
Res.		
9		
Res.		
8		
Res.		
7		
Res.		
6		
Res.		
5		
DBG_TIM19_STOP		
0		
4		
DBG_TIM17_STOP		
0		
3		
DBG_TIM16_STOP		
0		
2		
DBG_TIM15_STOP		
0		
1		
Res.		
0		
Res.		

1. The reset value is product dependent. For more information, refer to [Section 31.6.1: MCU device ID code](#).

## 32 Device electronic signature

The device electronic signature is stored in the System memory area of the Flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the STM32F37xx microcontroller.

### 32.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

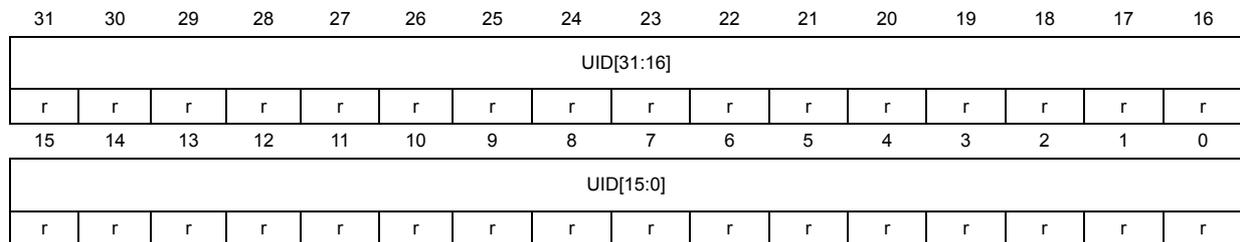
- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF F7AC

Address offset: 0x00

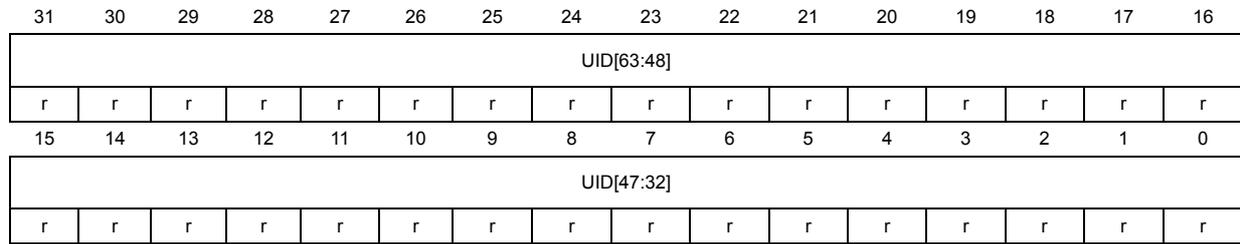
Read only = 0xXXXX XXXX where X is factory-programmed



Bits 31:0 **UID[31:0]**: X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

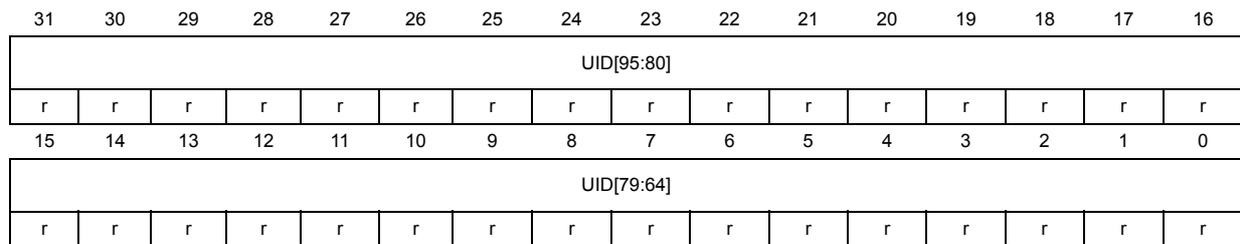


Bits 31:8 **UID[63:40]:** LOT\_NUM[23:0]  
 Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF\_NUM[7:0]  
 Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed



Bits 31:0 **UID[95:64]:** LOT\_NUM[55:24]  
 Lot number (ASCII encoded)

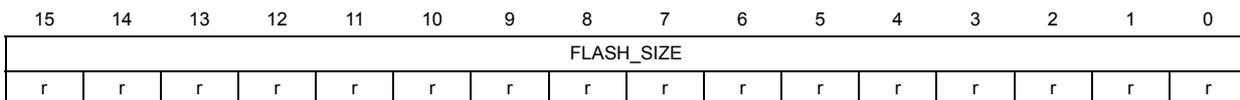
## 32.2 Memory size data register

### 32.2.1 Flash size data register

Base address: 0x1FFF F7CC

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed



Bits 15:0 **FLASH\_SIZE[15:0]:** Flash memory size  
 This bitfield indicates the size of the device Flash memory expressed in Kbytes.  
 As an example, 0x040 corresponds to 64 Kbytes.

# Index

## A

ADC_CR1	209
ADC_CR2	211
ADC_DR	220
ADC_HTR	215
ADC_JDRx	220
ADC_JOFRx	214
ADC_JSQR	219
ADC_LTR	215
ADC_SMPR1	213
ADC_SMPR2	214
ADC_SQR1	216
ADC_SQR2	217
ADC_SQR3	218
ADC_SR	208

## C

CAN_BTR	791
CAN_ESR	790
CAN_FA1R	801
CAN_FFA1R	801
CAN_FiRx	802
CAN_FM1R	800
CAN_FMR	799
CAN_FS1R	800
CAN_IER	788
CAN_MCR	781
CAN_MSR	783
CAN_RDHxR	798
CAN_RDLxR	798
CAN_RDTxR	797
CAN_RF0R	787
CAN_RF1R	787
CAN_RiRx	796
CAN_TDHxR	795
CAN_TDLxR	795
CAN_TDTxR	794
CAN_TiRx	793
CAN_TSR	784
CEC_CFGR	848
CEC_CR	847
CEC_IER	853
CEC_ISR	851
CEC_RXDR	851
CEC_TXDR	851

COMP_CSR	291
CRC_CR	72
CRC_DR	71
CRC_IDR	72
CRC_INIT	73
CRC_POL	73

## D

DAC_CR	276
DAC_DHR12L1	280
DAC_DHR12L2	281
DAC_DHR12LD	282
DAC_DHR12R1	279
DAC_DHR12R2	280
DAC_DHR12RD	282
DAC_DHR8R1	280
DAC_DHR8R2	281
DAC_DHR8RD	283
DAC_DOR1	283
DAC_DOR2	283
DAC_SR	284
DAC_SWTRIGR	279
DBGMCU_APB1_FZ	878
DBGMCU_APB2_FZ	880
DBGMCU_CR	876
DBGMCU_IDCODE	863
DMA_CCRx	176
DMA_CMARx	179
DMA_CNDTRx	178
DMA_CPARx	178
DMA_IFCR	175
DMA_ISR	174

## E

EXTI_EMR	191
EXTI_FTSTR	192
EXTI_IMR	191
EXTI_PR	193
EXTI_RTSTR	192
EXTI_SWIER	193

## F

FLASH_ACR	59
FLASH_CR	61
FLASH_KEYR	59
FLASH_OPTKEYR	60

FLASH\_SR ..... 60

## G

GPIOx\_AFRH ..... 151  
 GPIOx\_AFRL ..... 151  
 GPIOx\_BRR ..... 152  
 GPIOx\_BSRR ..... 149  
 GPIOx\_IDR ..... 148  
 GPIOx\_LCKR ..... 149  
 GPIOx\_MODER ..... 146  
 GPIOx\_ODR ..... 149  
 GPIOx\_OSPEEDR ..... 147  
 GPIOx\_OTYPER ..... 146  
 GPIOx\_PUPDR ..... 148

## I

I2C\_ISR ..... 614  
 I2Cx\_CR1 ..... 604  
 I2Cx\_CR2 ..... 607  
 I2Cx\_ICR ..... 616  
 I2Cx\_OAR1 ..... 610  
 I2Cx\_OAR2 ..... 611  
 I2Cx\_PECR ..... 617  
 I2Cx\_RXDR ..... 618  
 I2Cx\_TIMEOUTR ..... 613  
 I2Cx\_TIMINGR ..... 612  
 I2Cx\_TXDR ..... 618  
 IWDG\_KR ..... 494  
 IWDG\_PR ..... 495  
 IWDG\_RLR ..... 496  
 IWDG\_SR ..... 497  
 IWDG\_WINR ..... 498

## P

PWR\_CR ..... 89  
 PWR\_CSR ..... 91

## R

RCC\_AHBENR ..... 119  
 RCC\_AHBSTR ..... 130  
 RCC\_APB1ENR ..... 123  
 RCC\_APB1RSTR ..... 117  
 RCC\_APB2ENR ..... 121  
 RCC\_APB2RSTR ..... 115  
 RCC\_BDCR ..... 126  
 RCC\_CFGR ..... 109  
 RCC\_CFGR2 ..... 132  
 RCC\_CFGR3 ..... 133  
 RCC\_CIR ..... 113

RCC\_CR ..... 107  
 RCC\_CSR ..... 128  
 RTC\_ALRMAR ..... 534  
 RTC\_ALRMBR ..... 535  
 RTC\_ALRMBSSR ..... 547  
 RTC\_BKxR ..... 548  
 RTC\_CALR ..... 542  
 RTC\_CR ..... 526  
 RTC\_DR ..... 525  
 RTC\_ISR ..... 529  
 RTC\_PRER ..... 532  
 RTC\_SHIFTR ..... 538  
 RTC\_SSR ..... 537  
 RTC\_TAFCR ..... 543  
 RTC\_TCR ..... 543  
 RTC\_TR ..... 524  
 RTC\_TSDR ..... 540  
 RTC\_TSSSR ..... 541  
 RTC\_TSTR ..... 539  
 RTC\_WPR ..... 536  
 RTC\_WUTR ..... 533

## S

SDADC\_CLRISR ..... 247  
 SDADC\_CONF0R ..... 249  
 SDADC\_CONF1R ..... 250  
 SDADC\_CONF2R ..... 251  
 SDADC\_CONFCHR1 ..... 252  
 SDADC\_CONFCHR2 ..... 252  
 SDADC\_CR1 ..... 239  
 SDADC\_CR2 ..... 242  
 SDADC\_ISR ..... 245  
 SDADC\_JCHGR ..... 248  
 SDADC\_JDATA12R ..... 255  
 SDADC\_JDATA13R ..... 257  
 SDADC\_JDATAR ..... 253  
 SDADC\_RDATA13R ..... 258  
 SDADC\_RDATAR ..... 254  
 SPI\_CR1 ..... 730  
 SPI\_CR2 ..... 732  
 SPI\_CRCPR ..... 737  
 SPI\_DR ..... 737  
 SPI\_I2SCFGR ..... 739  
 SPI\_I2SPR ..... 741  
 SPI\_RXCR ..... 738  
 SPI\_TXCR ..... 738  
 SPIx\_CR2 ..... 732  
 SPIx\_SR ..... 735  
 SYSCFG\_EXTICR1 ..... 157  
 SYSCFG\_EXTICR2 ..... 158  
 SYSCFG\_EXTICR3 ..... 158

SYSCFG_EXTICR4 .....	159	TSC_IOSCR1 .....	758
SYSCFG_MEMRMP .....	155	TSC_ISR .....	756
<b>T</b>			
TIM14_OR .....	404	USART_BRR .....	673
TIM15_ARR .....	451	USART_CR1 .....	662
TIM15_BDTR .....	453	USART_CR2 .....	666
TIM15_CCER .....	448	USART_CR3 .....	670
TIM15_CCMR1 .....	445	USART_GTPR .....	673
TIM15_CCR1 .....	452	USART_ICR .....	681
TIM15_CCR2 .....	453	USART_ISR .....	677
TIM15_CNT .....	451	USART_RDR .....	683
TIM15_CR1 .....	437	USART_RQR .....	676
TIM15_CR2 .....	438	USART_RTOR .....	674
TIM15_DCR .....	455	USART_TDR .....	684
TIM15_DIER .....	442	USB_ADDRn_RX .....	833
TIM15_DMAR .....	456	USB_ADDRn_TX .....	832
TIM15_EGR .....	444	USB_BTABLE .....	827
TIM15_PSC .....	451	USB_CNTR .....	821
TIM15_RCR .....	452	USB_COUNTn_RX .....	834
TIM15_SMCR .....	440	USB_COUNTn_TX .....	833
TIM15_SR .....	443	USB_DADDR .....	827
TIM2_OR .....	355, 404	USB_EPnR .....	828
TIMx_ARR .....	350, 391, 402, 490	USB_FNR .....	826
TIMx_BDTR .....	473	USB_ISTR .....	823
TIMx_CCER .....	348, 390, 401, 468	<b>W</b>	
TIMx_CCMR1 .....	344, 387, 398, 465	WWDG_CFR .....	504
TIMx_CCMR2 .....	347	WWDG_CR .....	503
TIMx_CCR1 .....	351, 392, 403, 472	WWDG_SR .....	504
TIMx_CCR2 .....	351, 392		
TIMx_CCR3 .....	352		
TIMx_CCR4 .....	352		
TIMx_CNT .....	350, 391, 402, 471, 489		
TIMx_CR1 .....	335, 381, 395, 404, 459, 486		
TIMx_CR2 .....	337, 460, 488		
TIMx_DCR .....	353, 474		
TIMx_DIER .....	340, 383, 396, 462, 488		
TIMx_DMAR .....	353, 475		
TIMx_EGR .....	343, 386, 397, 464, 489		
TIMx_PSC .....	350, 391, 402, 471, 490		
TIMx_RCR .....	472		
TIMx_SMCR .....	338, 382		
TIMx_SR .....	341, 384, 396, 463, 489		
TSC_CR .....	752		
TSC_ICR .....	755		
TSC_IER .....	754		
TSC_IOASCR .....	757		
TSC_IOCCR1 .....	758		
TSC_IOGCSR .....	759		
TSC_IOGxCR .....	759		
TSC_IOHCR .....	756		

## 33 Revision history

**Table 139. Document revision history**

Date	Revision	Changes
05-Sep-2012	1	Initial release.
05-Dec-2013	2	<p>Documentation conventions: Updated <a href="#">Section 1.2: Glossary</a>.</p> <p>System architecture and memory overview: Updated <a href="#">Figure 1: System architecture</a> Added "(ICODE and DCODE)" for internal Flash memory in <a href="#">Section 2.1: System architecture</a></p> <p>Embedded Flash memory: Updated <a href="#">Section : Unlocking the Flash memory</a>, <a href="#">Table 4: Flash memory read protection status</a>, <a href="#">Table 7: Flash interface - register map and reset values</a>, <a href="#">Table 10: Description of the option bytes</a>. Renamed "FORCE_OPTLOAD" to "OBL_LAUNCH".</p> <p>PWR: Updated <a href="#">Figure 6: Power supply overview</a>. Updated Bit 3 in <a href="#">Section 6.4.2: Power control/status register (PWR_CSR)</a> and <a href="#">Section 6.4.3: PWR register map</a>. Added a note in <a href="#">Section : Entering Stop mode</a>. Updated <a href="#">Table 15: Stop mode</a>. Updated <a href="#">Section : Supply voltages</a>. Added a Caution note in <a href="#">Section 6.3.5: Standby mode</a>.</p> <p>RCC: Updated <a href="#">Figure 7.1: Reset</a> Updated <a href="#">Figure 10: Clock tree part 1</a> and <a href="#">Section 7.4.4: APB2 peripheral reset register (RCC_APB2RSTR)</a>, Bit 0. Renamed "FORCE_OBL" to "OBL_LAUNCH". Updated <a href="#">APB1 peripheral reset register (RCC_APB1RSTR)</a> and <a href="#">Section 7.4.14: RCC register map</a>.</p> <p>SYSCFG: Updated <a href="#">Section 9.1.2: SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)</a>, <a href="#">Section 9.1.3: SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)</a>, <a href="#">Section 9.1.4: SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)</a> and <a href="#">Section 9.1.5: SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)</a>.</p>

**Table 139. Document revision history (continued)**

Date	Revision	Changes
05-Dec-2013	2 (cont'd)	<p>DMA Updated <a href="#">Section : DMA1 controller</a> and <a href="#">Section : DMA2 controller</a>.</p> <p>Interrupts and events: Replaced reference to “Cortex<sup>®</sup>.M4” by “PM0214 programming manual” in <a href="#">Section 11.1.1: NVIC main features</a>. Updated <a href="#">Software interrupt event register (EXTI_SWIER)</a> and <a href="#">Pending register (EXTI_PR)</a>. Updated <a href="#">Figure 24: Extended interrupt/event GPIO mapping</a> and <a href="#">Table 29: Extended interrupt/event controller register map and reset values</a>.</p> <p>SDADC: Updated <a href="#">Figure 35: Switch configuration in single-ended mode</a>, <a href="#">Figure 36: Switch configuration in differential mode</a>, <a href="#">Figure 37: Switch configuration in mixed mode (example 1)</a> and <a href="#">Figure 38: Switch configuration in mixed mode (example 2)</a>. Replaced all "SDADC_CONFRx" by "SDADC_CONFxR" Updated <a href="#">Section 13.5.7: Launching calibration and determining the offset values</a>.</p> <p>DAC: Updated <a href="#">Figure 41: DAC1 block diagram</a> and <a href="#">Figure 42: DAC2 block diagram</a>: replaced “AEIC_9” by “EXT_9” Updated TSEL1 and TSEL2 description. Replaced “AIEC” by “EXTI”. Updated <a href="#">Table 42: DAC register map and reset values</a>.</p> <p>COMP: Updated <a href="#">Figure 51: Comparator 1 and 2 block diagrams</a>. Updated “COMP_INP_DAC” description. Updated <a href="#">Table 43: COMP register map and reset values</a>.</p> <p>IWDG: Updated <a href="#">Figure 162: Independent watchdog block diagram</a>.</p> <p>USART: Updated note in <a href="#">Section : Single byte communication</a>. Updated Mode 2 and Mode 3 description in <a href="#">Section 25.5.6: Auto baud rate detection</a> Removed note on bit 19(RWU) in <a href="#">Section 25.7.8: Interrupt &amp; status register (USART_ISR) on page 677</a>. Updated <a href="#">Section 25.5.10: LIN (local interconnection network) mode</a>. Updated <a href="#">Figure 197: USART block diagram</a> and added two notes. Replaced “BRR[3:0] = 0x3&lt;&lt;1=0x1” by “BRR[3:0] = 0x3&gt;&gt;1=0x1” in <a href="#">Section : How to derive USARTDIV from USART_BRR register values when OVER8=0</a>.</p>

**Table 139. Document revision history (continued)**

Date	Revision	Changes
05-Dec-2013	2 (cont'd)	<p>Replaced in Bit 2 MMRQ <a href="#">Section 25.7.7: Request register (USART_RQR)</a> “resets the RWU flag” by “sets the RWU flag”</p> <p>Added ‘In Smartcard, LIN and IrDA modes, only Oversampling by 16 is supported’ in <a href="#">Section 25.5.4: Baud rate generation</a></p> <p>Corrected and updated stop bits in <a href="#">Figure 198: Word length programming</a></p> <p>SPI/I2S: Updated <a href="#">Section 26.6: SPI interrupts</a> and <a href="#">Section 26.7.5: I2S slave mode</a>.</p> <p>TSC: Added <a href="#">Table 101: Interrupt control bits</a>. Replaced “Power-on reset value” with “Reset value” in <a href="#">Section 27.6.2: TSC interrupt enable register (TSC_IER)</a>.</p> <p>DBG: Updated <a href="#">Section 31.16.4: Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)</a>.</p> <p>General-purpose timers: Updated <a href="#">Section 18.1: TIM15/16/17 introduction</a>, <a href="#">Section 18.3: TIM16 and TIM17 main features</a>, <a href="#">Figure 123: TIM16 and TIM17 block diagram</a>, <a href="#">Section 17.4.3: Clock selection</a>, <a href="#">Section 18.6.1: TIM16&amp;TIM17 control register 1 (TIMx_CR1)</a>, <a href="#">Section 18.6.3: TIM16&amp;TIM17 DMA/interrupt enable register (TIMx_DIER)</a>, <a href="#">Section 18.6.4: TIM16&amp;TIM17 status register (TIMx_SR)</a>, <a href="#">Section 18.6.5: TIM16&amp;TIM17 event generation register (TIMx_EGR)</a> and <a href="#">Table 57: TIM16&amp;TIM17 register map and reset values</a>.</p> <p>I2C: Corrected <a href="#">Figure 184: Transfer sequence flowchart for I2C master transmitter for N&gt;255 bytes</a>. Removed maximum values of parameter “Data hold time” and added row “Data valid time” in <a href="#">Table 69: I2C-SMBUS specification data setup and hold times</a>. Updated sub-section <a href="#">I2C timings</a>. Updated <a href="#">Figure 168: Setup and hold timings</a>. Reclassified section “I2C register map” from <a href="#">2.8</a> to <a href="#">24.7.12</a>. Added <a href="#">Caution</a>: “If wakeup from Stop is disabled...” in <a href="#">Section 24.4.14: Wakeup from Stop mode on address match</a>. Added <a href="#">Section 24.5: I2C low-power modes</a>. Moved <a href="#">Section 24.7: I2C debug mode</a> to <a href="#">24.4.17</a> and renamed it <a href="#">Debug mode</a>.</p>

Table 139. Document revision history (continued)

Date	Revision	Changes
05-Dec-2013	2 (cont'd)	<p>Modified sub-section <i>Slave clock stretching (NOSTRETCH = 0)</i>.  Updated <i>Table 72: Examples of timings settings for fI2CCLK = 8 MHz</i>.  Updated <i>Figure 166: I2C block diagram</i>.</p> <p>RTC:  Replaced “power-on reset” with “backup domain reset” throughout <i>Section 23: Real-time clock (RTC)</i>.  Removed “the backup registers are reset when a tamper detection event occurs” in <i>Section 23.2: RTC main features</i>.  Updated <i>RTC backup registers (RTC_BKPxR)</i> and <i>RTC initialization and status register (RTC_ISR)</i> register.</p> <p>GPIO:  Updated <i>GPIO port output speed register (GPIOx_OSPEEDR) (x = A..F)</i>, <i>GPIO port input data register (GPIOx_IDR) (x = A..F)</i>, <i>GPIO port output data register (GPIOx_ODR) (x = A..F)</i> <i>GPIO port output type register (GPIOx_OTYPER) (x = A..F)</i>, <i>GPIO port mode register (GPIOx_MODER) (x = A..F)</i>, <i>GPIO port output speed register (GPIOx_OSPEEDR) (x = A..F)</i>, <i>GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..F)</i>, <i>GPIO register map</i></p>
07-May-2014	3	<p>Removed “STM32F38xx” in all the document. This declination of the product line is now called STM32F378xx.</p> <p>Replaced “Backup domain” by “RTC domain” in all the document except in <i>Section 6.1.2</i> title.</p> <p>Updated OBL_LAUNCH bit description in <i>Section 3.5.5: Flash control register (FLASH_CR)</i>.</p> <p>Updated <i>Section 5.2: CRC main features</i>.</p> <p>Replaced VDDA_MON by VDDA_MONITOR in <i>Section 3.5.7: Option byte register (FLASH_OBR)</i>, <i>Table 7: Flash interface - register map and reset values</i>, and <i>Table 10: Description of the option bytes</i>.</p> <p>Updated <i>Section 6.1.2: Battery backup domain</i> and <i>Section 6.2.1: Power on reset (POR)/power down reset (PDR)</i>.</p> <p>Updated <i>Table 19: Port bit configuration table</i>.</p> <p>Updated <i>Section 8.3.7: I/O alternate function input/output</i>, <i>Section 8.3.8: External interrupt/wakeup lines</i> and <i>Section 8.3.12: Analog configuration</i>.</p> <p>Updated bit IDRx description in <i>Section 8.4.5: GPIO port input data register (GPIOx_IDR) (x = A..F)</i>.</p> <p>Updated bit ODRx description in <i>Section 8.4.6: GPIO port output data register (GPIOx_ODR) (x = A..F)</i>.</p> <p>Updated <i>Section 9.1.1: SYSCFG configuration register 1 (SYSCFG_CFGR1)</i>.</p> <p>Updated last bullet in <i>Section 10.2: DMA main features</i>.</p> <p>Updated <i>Section 12.10: Battery voltage monitoring</i>.</p>

**Table 139. Document revision history (continued)**

Date	Revision	Changes
07-May-2014	3 (cont'd)	<p>Updated <a href="#">Section 14.2: DAC1/2 main features</a>, <a href="#">Section 14.3.4: DAC channel conversion</a>, <a href="#">Section 14.4.4: DAC channel conversion in dual mode</a>, <a href="#">Section 14.4.5: Description of dual conversion modes</a>, <a href="#">Section 14.4.7: DAC trigger selection</a>, <a href="#">Section 14.5: Noise generation</a>, <a href="#">Section 14.6: Triangle-wave generation</a>, MAMP2, WAVE2, MAMP1 and WAVE1 bit descriptions in <a href="#">Section 14.8.1: DAC control register (DAC_CR)</a>.</p> <p>Updated <a href="#">Section 15.1: Introduction</a>, <a href="#">Section 15.3.4: Comparator LOCK mechanism</a>.</p> <p>Replaced “TIM16” by “TIM16_OC1” and “TIM17” by “TIM17_OC1” in <a href="#">Table 53: TIMx Internal trigger connection</a>.</p> <p>Updated IC1F bit description in <a href="#">Section 16.4.7: TIMx capture/compare mode register 1 (TIMx_CCMR1)</a>.</p> <p>Updated <a href="#">Table 88: Frame formats</a> and MSBFIRST bit description in <a href="#">Section 25.7.2: Control register 2 (USART_CR2)</a>.</p> <p>Updated <a href="#">Section 19: Infrared interface (IRTIM)</a> and <a href="#">Figure 151: IR internal hardware connections with TIM16 and TIM17</a>.</p> <p>Updated <a href="#">Section 21.3.1: IWDG block diagram</a>, <a href="#">Section 21.3.5: Debug mode</a>, PR bit description in <a href="#">Section 21.4.1: Key register (IWDG_KR)</a>.</p> <p>Updated <a href="#">Section 23.3.1: RTC block diagram</a>, <a href="#">Section 23.3.4: Real-time clock and calendar</a>, <a href="#">Section 23.3.12: RTC smooth digital calibration</a>.</p> <p>Updated <a href="#">Section 23.5: RTC interrupts</a>.</p> <p>Updated <a href="#">Section 24.4.9: I2Cx_TIMINGR register configuration examples</a>, added access type to all register descriptions in <a href="#">Section 24.7: I2C registers</a>.</p> <p>Updated <a href="#">Note...</a> Updated CTSE bit description in <a href="#">Section 25.7.3: Control register 3 (USART_CR3)</a>.</p> <p>Updated <a href="#">Section 26.2: SPI main features</a>, moved <a href="#">Section 26.7.8: DMA features</a>, updated bit LSBFIRST description in <a href="#">Section 26.9.1: SPI control register 1 (SPIx_CR1)</a>, updated reset value in <a href="#">Section 26.9.2: SPI control register 2 (SPIx_CR2)</a>.</p> <p>Updated <a href="#">Section Table 99.: Capacitive sensing GPIOs available on STM32F378xx devices</a>.</p> <p>Added "512 Bytes of dedicated packet buffer memory SRAM" in <a href="#">Section 29.2: USB main features</a>.</p> <p>Added <a href="#">Section 29.3: USB implementation</a>, updated <a href="#">Section 29.4: USB functional description</a>.</p> <p>Updated <a href="#">Table 106: STM32F37xx USB implementation</a>.</p> <p>Updated <a href="#">Section 30.4.1: SFT option bit</a>, <a href="#">Section 30.5.4: Short Bit Period Error (SBPE)</a>, bit OAR in <a href="#">Section 30.7.2: CEC configuration register (CEC_CFGR)</a>, bit RXACKE in <a href="#">Section 30.7.5: CEC Interrupt and Status Register (CEC_ISR)</a>.</p> <p>Updated <a href="#">Section 31.4.4: Using serial wire and releasing the unused debug pins as GPIOs</a>.</p>

**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.**

**ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.**

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.

© 2014 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](http://www.st.com)