

## Introduction

This reference manual complements the datasheets of the STM32C0x1 microcontrollers, providing information required for application and in particular for software development. It pertains to the superset of feature sets available on STM32C0x1 microcontrollers.

For feature set, ordering information, and mechanical and electrical characteristics of a particular STM32C0x1 device, refer to its corresponding datasheet.

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

## Related documents

- “*Cortex®-M0+ Technical Reference Manual*”, available from: <http://infocenter.arm.com>
- PM0223 programming manual for Cortex®-M0+ core<sup>(a)</sup>
- STM32C0x1 datasheets<sup>(a)</sup>
- AN2606 application note on booting STM32 MCUs<sup>(a)</sup>

---

a. Available on STMicroelectronics website [www.st.com](http://www.st.com)

# Contents

<b>1</b>	<b>Documentation conventions</b>	<b>35</b>
1.1	General information	35
1.2	List of abbreviations for registers	35
1.3	Glossary	36
1.4	Availability of peripherals	36
<b>2</b>	<b>Memory and bus architecture</b>	<b>37</b>
2.1	System architecture	37
2.2	Memory organization	39
2.2.1	Introduction	39
2.2.2	Memory map and register boundary addresses	40
2.3	Embedded SRAM	43
2.4	Flash memory overview	44
2.5	Boot configuration	44
<b>3</b>	<b>Embedded flash memory (FLASH)</b>	<b>47</b>
3.1	FLASH Introduction	47
3.2	FLASH main features	47
3.3	FLASH functional description	47
3.3.1	Flash memory organization	47
3.3.2	FLASH empty check	48
3.3.3	FLASH read access latency	48
3.3.4	Flash memory acceleration	49
3.3.5	FLASH program and erase operations	50
3.3.6	FLASH Main memory erase sequences	51
3.3.7	FLASH Main memory programming sequences	52
3.4	FLASH option bytes	55
3.4.1	FLASH option byte description	55
3.4.2	FLASH option byte programming	57
3.5	Flash memory protection	58
3.5.1	FLASH read protection (RDP)	58
3.5.2	FLASH proprietary code readout protection (PCROP)	61
3.5.3	FLASH write protection (WRP)	63

3.5.4	Securable memory area . . . . .	64
3.5.5	Disabling core debug access . . . . .	65
3.5.6	Forcing boot from flash memory . . . . .	65
3.6	FLASH interrupts . . . . .	65
3.7	FLASH registers . . . . .	67
3.7.1	FLASH access control register (FLASH_ACR) . . . . .	67
3.7.2	FLASH key register (FLASH_KEYR) . . . . .	68
3.7.3	FLASH option key register (FLASH_OPTKEYR) . . . . .	68
3.7.4	FLASH status register (FLASH_SR) . . . . .	69
3.7.5	FLASH control register (FLASH_CR) . . . . .	70
3.7.6	FLASH option register (FLASH_OPTR) . . . . .	72
3.7.7	FLASH PCROP area A start address register (FLASH_PCROP1ASR) . . . . .	75
3.7.8	FLASH PCROP area A end address register (FLASH_PCROP1AER) . . . . .	75
3.7.9	FLASH WRP area A address register (FLASH_WRP1AR) . . . . .	76
3.7.10	FLASH WRP area B address register (FLASH_WRP1BR) . . . . .	77
3.7.11	FLASH PCROP area B start address register (FLASH_PCROP1BSR) . . . . .	77
3.7.12	FLASH PCROP area B end address register (FLASH_PCROP1BER) . . . . .	78
3.7.13	FLASH security register (FLASH_SECR) . . . . .	78
3.7.14	FLASH register map . . . . .	79
<b>4</b>	<b>Power control (PWR) . . . . .</b>	<b>81</b>
4.1	Power supplies and voltage references . . . . .	81
4.1.1	ADC reference voltage . . . . .	81
4.1.2	Voltage regulator . . . . .	81
4.2	Power supply supervisor . . . . .	82
4.2.1	Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR) . . . . .	82
4.3	Operating modes . . . . .	84
4.3.1	Power saving in run mode . . . . .	86
4.3.2	Low-power modes . . . . .	86
4.4	PWR registers . . . . .	90
4.4.1	PWR control register 1 (PWR_CR1) . . . . .	90
4.4.2	PWR control register 3 (PWR_CR3) . . . . .	91
4.4.3	PWR control register 4 (PWR_CR4) . . . . .	92

---

4.4.4	PWR status register 1 (PWR_SR1) . . . . .	93
4.4.5	PWR status register 2 (PWR_SR2) . . . . .	94
4.4.6	PWR status clear register (PWR_SCR) . . . . .	95
4.4.7	PWR Port A pull-up control register (PWR_PUCRA) . . . . .	95
4.4.8	PWR Port A pull-down control register (PWR_PDCRA) . . . . .	96
4.4.9	PWR Port B pull-up control register (PWR_PUCRB) . . . . .	97
4.4.10	PWR Port B pull-down control register (PWR_PDCRB) . . . . .	97
4.4.11	PWR Port C pull-up control register (PWR_PUCRC) . . . . .	98
4.4.12	PWR Port C pull-down control register (PWR_PDCRC) . . . . .	98
4.4.13	PWR Port D pull-up control register (PWR_PUCRD) . . . . .	99
4.4.14	PWR Port D pull-down control register (PWR_PDCRD) . . . . .	99
4.4.15	PWR Port F pull-up control register (PWR_PUCRF) . . . . .	100
4.4.16	PWR Port F pull-down control register (PWR_PDCRF) . . . . .	100
4.4.17	PWR backup x register (PWR_BKPxR) . . . . .	101
4.4.18	PWR register map . . . . .	101
<b>5</b>	<b>Reset and clock control (RCC) . . . . .</b>	<b>103</b>
5.1	Reset . . . . .	103
5.1.1	Power reset . . . . .	103
5.1.2	System reset . . . . .	103
5.1.3	RTC domain reset . . . . .	105
5.2	Clocks . . . . .	105
5.2.1	HSE clock . . . . .	108
5.2.2	HSI48 clock . . . . .	110
5.2.3	LSE clock . . . . .	110
5.2.4	LSI clock . . . . .	111
5.2.5	System clock (SYSCLK) selection . . . . .	111
5.2.6	Clock security system (CSS) . . . . .	111
5.2.7	Clock security system for LSE clock (LSECSS) . . . . .	112
5.2.8	ADC clock . . . . .	112
5.2.9	RTC clock . . . . .	112
5.2.10	Timer clock . . . . .	113
5.2.11	Watchdog clock . . . . .	113
5.2.12	Clock-out capability . . . . .	113
5.2.13	Internal/external clock measurement with TIM14/TIM16/TIM17 . . . . .	114
5.2.14	Peripheral clock enable registers . . . . .	116
5.3	Low-power modes . . . . .	116

5.4	RCC registers . . . . .	117
5.4.1	RCC clock control register (RCC_CR) . . . . .	117
5.4.2	RCC internal clock source calibration register (RCC_ICSCR) . . . . .	119
5.4.3	RCC clock configuration register (RCC_CFGR) . . . . .	120
5.4.4	RCC clock interrupt enable register (RCC_CIER) . . . . .	122
5.4.5	RCC clock interrupt flag register (RCC_CIFR) . . . . .	123
5.4.6	RCC clock interrupt clear register (RCC_CICR) . . . . .	124
5.4.7	RCC I/O port reset register (RCC_IOPRSTR) . . . . .	125
5.4.8	RCC AHB peripheral reset register (RCC_AHBRSTR) . . . . .	126
5.4.9	RCC APB peripheral reset register 1 (RCC_APBRSTR1) . . . . .	126
5.4.10	RCC APB peripheral reset register 2 (RCC_APBRSTR2) . . . . .	127
5.4.11	RCC I/O port clock enable register (RCC_IOPENR) . . . . .	129
5.4.12	RCC AHB peripheral clock enable register (RCC_AHBENR) . . . . .	130
5.4.13	RCC APB peripheral clock enable register 1 (RCC_APBENR1) . . . . .	130
5.4.14	RCC APB peripheral clock enable register 2 (RCC_APBENR2) . . . . .	132
5.4.15	RCC I/O port in Sleep mode clock enable register (RCC_IOPSMENR) . . . . .	133
5.4.16	RCC AHB peripheral clock enable in Sleep/Stop mode register (RCC_AHBSMENR) . . . . .	134
5.4.17	RCC APB peripheral clock enable in Sleep/Stop mode register 1 (RCC_APBSMENR1) . . . . .	134
5.4.18	RCC APB peripheral clock enable in Sleep/Stop mode register 2 (RCC_APBSMENR2) . . . . .	136
5.4.19	RCC peripherals independent clock configuration register (RCC_CCIPR)	
137		
5.4.20	RCC control/status register 1 (RCC_CSR1) . . . . .	138
5.4.21	RCC control/status register 2 (RCC_CSR2) . . . . .	140
5.4.22	RCC register map . . . . .	142
<b>6</b>	<b>General-purpose I/Os (GPIO) . . . . .</b>	<b>145</b>
6.1	Introduction . . . . .	145
6.2	GPIO main features . . . . .	145
6.3	GPIO functional description . . . . .	145
6.3.1	General-purpose I/O (GPIO) . . . . .	147
6.3.2	I/O pin alternate function multiplexer and mapping . . . . .	148
6.3.3	I/O port control registers . . . . .	148
6.3.4	I/O port data registers . . . . .	148
6.3.5	I/O data bitwise handling . . . . .	148

---

6.3.6	GPIO locking mechanism . . . . .	149
6.3.7	I/O alternate function input/output . . . . .	149
6.3.8	External interrupt/wakeup lines . . . . .	149
6.3.9	Input configuration . . . . .	149
6.3.10	Output configuration . . . . .	150
6.3.11	Alternate function configuration . . . . .	151
6.3.12	Analog configuration . . . . .	152
6.3.13	Using the HSE or LSE oscillator pins as GPIOs . . . . .	153
6.3.14	Small package adjustment . . . . .	153
6.3.15	Reset pin (PF2) in GPIO mode . . . . .	153
6.3.16	Boot0 pin (PA14) in GPIO mode . . . . .	153
6.4	GPIO registers . . . . .	154
6.4.1	GPIO port mode register (GPIOx_MODER) (x = A, B, C, D, F) . . . . .	154
6.4.2	GPIO port output type register (GPIOx_OTYPER) (x = A, B, C, D, F) . . . . .	154
6.4.3	GPIO port output speed register (GPIOx_OSPEEDR) (x = A, B, C, D, F) . . . . .	155
6.4.4	GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A, B, C, D, F) . . . . .	155
6.4.5	GPIO port input data register (GPIOx_IDR) (x = A, B, C, D, F) . . . . .	156
6.4.6	GPIO port output data register (GPIOx_ODR) (x = A, B, C, D, F) . . . . .	156
6.4.7	GPIO port bit set/reset register (GPIOx_BSRR) (x = A, B, C, D, F) . . . . .	156
6.4.8	GPIO port configuration lock register (GPIOx_LCKR) (x = A, B, C, D, F) . . . . .	157
6.4.9	GPIO alternate function low register (GPIOx_AFRL) (x = A, B, C, D, F) . . . . .	158
6.4.10	GPIO alternate function high register (GPIOx_AFRH) (x = A, B, C, D, F) . . . . .	159
6.4.11	GPIO port bit reset register (GPIOx_BRR) (x = A, B, C, D, F) . . . . .	159
6.4.12	GPIO register map . . . . .	160
7	System configuration controller (SYSCFG) . . . . .	161
7.1	SYSCFG registers . . . . .	161
7.1.1	SYSCFG configuration register 1 (SYSCFG_CFGR1) . . . . .	161
7.1.2	SYSCFG configuration register 2 (SYSCFG_CFGR2) . . . . .	163
7.1.3	SYSCFG configuration register 3 (SYSCFG_CFGR3) . . . . .	164

---

7.1.4	SYSCFG interrupt line 0 status register (SYSCFG_ITLINE0) . . . . .	165
7.1.5	SYSCFG interrupt line 2 status register (SYSCFG_ITLINE2) . . . . .	166
7.1.6	SYSCFG interrupt line 3 status register (SYSCFG_ITLINE3) . . . . .	166
7.1.7	SYSCFG interrupt line 4 status register (SYSCFG_ITLINE4) . . . . .	167
7.1.8	SYSCFG interrupt line 5 status register (SYSCFG_ITLINE5) . . . . .	168
7.1.9	SYSCFG interrupt line 6 status register (SYSCFG_ITLINE6) . . . . .	168
7.1.10	SYSCFG interrupt line 7 status register (SYSCFG_ITLINE7) . . . . .	168
7.1.11	SYSCFG interrupt line 9 status register (SYSCFG_ITLINE9) . . . . .	169
7.1.12	SYSCFG interrupt line 10 status register (SYSCFG_ITLINE10) . . . . .	169
7.1.13	SYSCFG interrupt line 11 status register (SYSCFG_ITLINE11) . . . . .	170
7.1.14	SYSCFG interrupt line 12 status register (SYSCFG_ITLINE12) . . . . .	170
7.1.15	SYSCFG interrupt line 13 status register (SYSCFG_ITLINE13) . . . . .	170
7.1.16	SYSCFG interrupt line 14 status register (SYSCFG_ITLINE14) . . . . .	171
7.1.17	SYSCFG interrupt line 16 status register (SYSCFG_ITLINE16) . . . . .	171
7.1.18	SYSCFG interrupt line 19 status register (SYSCFG_ITLINE19) . . . . .	171
7.1.19	SYSCFG interrupt line 21 status register (SYSCFG_ITLINE21) . . . . .	172
7.1.20	SYSCFG interrupt line 22 status register (SYSCFG_ITLINE22) . . . . .	172
7.1.21	SYSCFG interrupt line 23 status register (SYSCFG_ITLINE23) . . . . .	172
7.1.22	SYSCFG interrupt line 25 status register (SYSCFG_ITLINE25) . . . . .	173
7.1.23	SYSCFG interrupt line 27 status register (SYSCFG_ITLINE27) . . . . .	173
7.1.24	SYSCFG interrupt line 28 status register (SYSCFG_ITLINE28) . . . . .	173
7.1.25	SYSCFG register map . . . . .	174
<b>8</b>	<b>Interconnect matrix . . . . .</b>	<b>177</b>
8.1	Introduction . . . . .	177
8.2	Connection summary . . . . .	177
8.3	Interconnection details . . . . .	178
8.3.1	From TIM1, TIM3, TIM14, and TIM17, to TIM1 and TIM3 . . . . .	178
8.3.2	From TIM1, TIM3, and EXTI, to ADC . . . . .	179
8.3.3	From ADC to TIM1 . . . . .	179
8.3.4	From HSE, LSE, LSI, MCO, MCO2, and RTC, to TIM14, TIM16, and TIM17 . . . . .	179
8.3.5	From internal analog sources to ADC . . . . .	180
8.3.6	From system errors to TIM1, TIM16, and TIM17 . . . . .	180
8.3.7	From TIM16, TIM17, USART1, and USART2, to IRTIM . . . . .	180
8.3.8	From TIM14 and EXTI to DMAMUX . . . . .	181

---

<b>9</b>	<b>Direct memory access controller (DMA) . . . . .</b>	<b>182</b>
9.1	Introduction . . . . .	182
9.2	DMA main features . . . . .	182
9.3	DMA implementation . . . . .	183
9.3.1	DMA1 . . . . .	183
9.3.2	DMA request mapping . . . . .	183
9.4	DMA functional description . . . . .	183
9.4.1	DMA block diagram . . . . .	183
9.4.2	DMA pins and internal signals . . . . .	184
9.4.3	DMA transfers . . . . .	184
9.4.4	DMA arbitration . . . . .	185
9.4.5	DMA channels . . . . .	185
9.4.6	DMA data width, alignment and endianness . . . . .	189
9.4.7	DMA error management . . . . .	190
9.5	DMA interrupts . . . . .	191
9.6	DMA registers . . . . .	191
9.6.1	DMA interrupt status register (DMA_ISR) . . . . .	191
9.6.2	DMA interrupt flag clear register (DMA_IFCR) . . . . .	192
9.6.3	DMA channel x configuration register (DMA_CCRx) . . . . .	193
9.6.4	DMA channel x number of data to transfer register (DMA_CNDTRx) . . . . .	195
9.6.5	DMA channel x peripheral address register (DMA_CPARx) . . . . .	196
9.6.6	DMA channel x memory address register (DMA_CMARx) . . . . .	197
9.6.7	DMA register map . . . . .	197
<b>10</b>	<b>DMA request multiplexer (DMAMUX) . . . . .</b>	<b>199</b>
10.1	Introduction . . . . .	199
10.2	DMAMUX main features . . . . .	200
10.3	DMAMUX implementation . . . . .	200
10.3.1	DMAMUX instantiation . . . . .	200
10.3.2	DMAMUX mapping . . . . .	200
10.4	DMAMUX functional description . . . . .	203
10.4.1	DMAMUX block diagram . . . . .	203
10.4.2	DMAMUX signals . . . . .	204
10.4.3	DMAMUX channels . . . . .	204
10.4.4	DMAMUX request line multiplexer . . . . .	204
10.4.5	DMAMUX request generator . . . . .	207

10.5	DMAMUX interrupts . . . . .	208
10.6	DMAMUX registers . . . . .	209
10.6.1	DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) . . . . .	209
10.6.2	DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) . . . . .	210
10.6.3	DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) . . . . .	210
10.6.4	DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) . . . . .	211
10.6.5	DMAMUX request generator interrupt status register (DMAMUX_RGSR) . . . . .	212
10.6.6	DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) . . . . .	212
10.6.7	DMAMUX register map . . . . .	213
<b>11</b>	<b>Nested vectored interrupt controller (NVIC) . . . . .</b>	<b>214</b>
11.1	Main features . . . . .	214
11.2	SysTick calibration value register . . . . .	214
11.3	Interrupt and exception vectors . . . . .	214
<b>12</b>	<b>Extended interrupt and event controller (EXTI) . . . . .</b>	<b>217</b>
12.1	EXTI main features . . . . .	217
12.2	EXTI block diagram . . . . .	217
12.2.1	EXTI connections between peripherals and CPU . . . . .	219
12.3	EXTI functional description . . . . .	219
12.3.1	EXTI configurable event input wakeup . . . . .	220
12.3.2	EXTI direct event input wakeup . . . . .	220
12.3.3	EXTI multiplexer . . . . .	221
12.4	EXTI functional behavior . . . . .	222
12.5	EXTI registers . . . . .	223
12.5.1	EXTI rising trigger selection register (EXTI_RTSR1) . . . . .	223
12.5.2	EXTI falling trigger selection register 1 (EXTI_FTSR1) . . . . .	224
12.5.3	EXTI software interrupt event register 1 (EXTI_SWIER1) . . . . .	224
12.5.4	EXTI rising edge pending register 1 (EXTI_RPR1) . . . . .	225
12.5.5	EXTI falling edge pending register 1 (EXTI_FPR1) . . . . .	225
12.5.6	EXTI external interrupt selection register (EXTI_EXTICRx) . . . . .	226
12.5.7	EXTI CPU wakeup with interrupt mask register (EXTI_IMR1) . . . . .	227

---

12.5.8	EXTI CPU wakeup with event mask register (EXTI_EMR1) . . . . .	228
12.5.9	EXTI register map . . . . .	229
<b>13</b>	<b>Cyclic redundancy check calculation unit (CRC) . . . . .</b>	<b>231</b>
13.1	Introduction . . . . .	231
13.2	CRC main features . . . . .	231
13.3	CRC functional description . . . . .	232
13.3.1	CRC block diagram . . . . .	232
13.3.2	CRC internal signals . . . . .	232
13.3.3	CRC operation . . . . .	232
13.4	CRC registers . . . . .	234
13.4.1	CRC data register (CRC_DR) . . . . .	234
13.4.2	CRC independent data register (CRC_IDR) . . . . .	234
13.4.3	CRC control register (CRC_CR) . . . . .	235
13.4.4	CRC initial value (CRC_INIT) . . . . .	236
13.4.5	CRC polynomial (CRC_POL) . . . . .	236
13.4.6	CRC register map . . . . .	237
<b>14</b>	<b>Analog-to-digital converter (ADC) . . . . .</b>	<b>238</b>
14.1	Introduction . . . . .	238
14.2	ADC main features . . . . .	239
14.3	ADC implementation . . . . .	240
14.4	ADC functional description . . . . .	241
14.4.1	ADC pins and internal signals . . . . .	241
14.4.2	ADC voltage regulator (ADVREGEN) . . . . .	242
14.4.3	Calibration (ADCAL) . . . . .	243
14.4.4	ADC on-off control (ADEN, ADDIS, ADRDY) . . . . .	244
14.4.5	ADC clock (CKMODE, PRESC[3:0]) . . . . .	246
14.4.6	ADC connectivity . . . . .	248
14.4.7	Configuring the ADC . . . . .	249
14.4.8	Channel selection (CHSEL, SCANDIR, CHSELRMOD) . . . . .	249
14.4.9	Programmable sampling time (SMPx[2:0]) . . . . .	250
14.4.10	Single conversion mode (CONT = 0) . . . . .	251
14.4.11	Continuous conversion mode (CONT = 1) . . . . .	251
14.4.12	Starting conversions (ADSTART) . . . . .	252
14.4.13	Timings . . . . .	253

14.4.14	Stopping an ongoing conversion (ADSTP) . . . . .	254
14.5	Conversion on external trigger and trigger polarity (EXTSEL, EXTEN) . . . . .	254
14.5.1	Discontinuous mode (DISCEN) . . . . .	255
14.5.2	Programmable resolution (RES) - Fast conversion mode . . . . .	255
14.5.3	End of conversion, end of sampling phase (EOC, EOSMP flags) . . . . .	256
14.5.4	End of conversion sequence (EOS flag) . . . . .	256
14.5.5	Example timing diagrams (single/continuous modes hardware/software triggers) . . . . .	257
14.5.6	Low frequency trigger mode . . . . .	259
14.6	Data management . . . . .	259
14.6.1	Data register and data alignment (ADC_DR, ALIGN) . . . . .	259
14.6.2	ADC overrun (OVR, OVRMOD) . . . . .	259
14.6.3	Managing a sequence of data converted without using the DMA . . . . .	261
14.6.4	Managing converted data without using the DMA without overrun . . . . .	261
14.6.5	Managing converted data using the DMA . . . . .	261
14.7	Low-power features . . . . .	262
14.7.1	Wait mode conversion . . . . .	262
14.7.2	Auto-off mode (AUTOFF) . . . . .	263
14.8	Analog window watchdogs . . . . .	265
14.8.1	Description of analog watchdog 1 . . . . .	265
14.8.2	Description of analog watchdog 2 and 3 . . . . .	266
14.8.3	ADC_AWDx_OUT output signal generation . . . . .	266
14.8.4	Analog Watchdog threshold control . . . . .	268
14.9	Oversampler . . . . .	269
14.9.1	ADC operating modes supported when oversampling . . . . .	271
14.9.2	Analog watchdog . . . . .	271
14.9.3	Triggered mode . . . . .	271
14.10	Temperature sensor and internal reference voltage . . . . .	272
14.11	ADC interrupts . . . . .	275
14.12	ADC registers . . . . .	276
14.12.1	ADC interrupt and status register (ADC_ISR) . . . . .	276
14.12.2	ADC interrupt enable register (ADC_IER) . . . . .	278
14.12.3	ADC control register (ADC_CR) . . . . .	280
14.12.4	ADC configuration register 1 (ADC_CFGR1) . . . . .	282
14.12.5	ADC configuration register 2 (ADC_CFGR2) . . . . .	285
14.12.6	ADC sampling time register (ADC_SMPR) . . . . .	286

---

14.12.7	ADC watchdog threshold register (ADC_AWD1TR) . . . . .	287
14.12.8	ADC watchdog threshold register (ADC_AWD2TR) . . . . .	288
14.12.9	ADC channel selection register (ADC_CHSELR) . . . . .	289
14.12.10	ADC channel selection register [alternate] (ADC_CHSELR) . . . . .	290
14.12.11	ADC watchdog threshold register (ADC_AWD3TR) . . . . .	292
14.12.12	ADC data register (ADC_DR) . . . . .	292
14.12.13	ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . . .	293
14.12.14	ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR) . . . . .	293
14.12.15	ADC calibration factor (ADC_CALFACT) . . . . .	294
14.12.16	ADC common configuration register (ADC_CCR) . . . . .	294
14.13	ADC register map . . . . .	295
<b>15</b>	<b>Advanced-control timer (TIM1) . . . . .</b>	<b>297</b>
15.1	TIM1 introduction . . . . .	297
15.2	TIM1 main features . . . . .	298
15.3	TIM1 functional description . . . . .	300
15.3.1	Time-base unit . . . . .	300
15.3.2	Counter modes . . . . .	302
15.3.3	Repetition counter . . . . .	313
15.3.4	External trigger input . . . . .	315
15.3.5	Clock selection . . . . .	316
15.3.6	Capture/compare channels . . . . .	320
15.3.7	Input capture mode . . . . .	322
15.3.8	PWM input mode . . . . .	323
15.3.9	Forced output mode . . . . .	324
15.3.10	Output compare mode . . . . .	325
15.3.11	PWM mode . . . . .	326
15.3.12	Asymmetric PWM mode . . . . .	329
15.3.13	Combined PWM mode . . . . .	330
15.3.14	Combined 3-phase PWM mode . . . . .	331
15.3.15	Complementary outputs and dead-time insertion . . . . .	332
15.3.16	Using the break function . . . . .	334
15.3.17	Bidirectional break inputs . . . . .	340
15.3.18	Clearing the OCxREF signal on an external event . . . . .	342
15.3.19	6-step PWM generation . . . . .	343
15.3.20	One-pulse mode . . . . .	344
15.3.21	Retriggerable one pulse mode . . . . .	345

15.3.22	Encoder interface mode . . . . .	346
15.3.23	UIF bit remapping . . . . .	348
15.3.24	Timer input XOR function . . . . .	349
15.3.25	Interfacing with Hall sensors . . . . .	349
15.3.26	Timer synchronization . . . . .	352
15.3.27	ADC synchronization . . . . .	356
15.3.28	DMA burst mode . . . . .	356
15.3.29	Debug mode . . . . .	357
15.4	TIM1 registers . . . . .	358
15.4.1	TIM1 control register 1 (TIM1_CR1) . . . . .	358
15.4.2	TIM1 control register 2 (TIM1_CR2) . . . . .	359
15.4.3	TIM1 slave mode control register (TIM1_SMCR) . . . . .	362
15.4.4	TIM1 DMA/interrupt enable register (TIM1_DIER) . . . . .	364
15.4.5	TIM1 status register (TIM1_SR) . . . . .	366
15.4.6	TIM1 event generation register (TIM1_EGR) . . . . .	368
15.4.7	TIM1 capture/compare mode register 1(TIM1_CCMR1) . . . . .	369
15.4.8	TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) . . . . .	370
15.4.9	TIM1 capture/compare mode register 2 (TIM1_CCMR2) . . . . .	373
15.4.10	TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) . . . . .	374
15.4.11	TIM1 capture/compare enable register (TIM1_CCER) . . . . .	376
15.4.12	TIM1 counter (TIM1_CNT) . . . . .	379
15.4.13	TIM1 prescaler (TIM1_PSC) . . . . .	379
15.4.14	TIM1 auto-reload register (TIM1_ARR) . . . . .	379
15.4.15	TIM1 repetition counter register (TIM1_RCR) . . . . .	380
15.4.16	TIM1 capture/compare register 1 (TIM1_CCR1) . . . . .	380
15.4.17	TIM1 capture/compare register 2 (TIM1_CCR2) . . . . .	381
15.4.18	TIM1 capture/compare register 3 (TIM1_CCR3) . . . . .	381
15.4.19	TIM1 capture/compare register 4 (TIM1_CCR4) . . . . .	382
15.4.20	TIM1 break and dead-time register (TIM1_BDTR) . . . . .	382

---

15.4.21	TIM1 DMA control register (TIM1_DCR) . . . . .	386
15.4.22	TIM1 DMA address for full transfer (TIM1_DMAR) . . . . .	387
15.4.23	TIM1 capture/compare mode register 3 (TIM1_CCMR3) . . . . .	388
15.4.24	TIM1 capture/compare register 5 (TIM1_CCR5) . . . . .	389
15.4.25	TIM1 capture/compare register 6 (TIM1_CCR6) . . . . .	390
15.4.26	TIM1 alternate function option register 1 (TIM1_AF1) . . . . .	390
15.4.27	TIM1 Alternate function register 2 (TIM1_AF2) . . . . .	391
15.4.28	TIM1 timer input selection register (TIM1_TISEL) . . . . .	392
15.4.29	TIM1 register map . . . . .	393
<b>16</b>	<b>General-purpose timer (TIM3) . . . . .</b>	<b>396</b>
16.1	TIM3 introduction . . . . .	396
16.2	TIM3 main features . . . . .	396
16.3	TIM3 functional description . . . . .	398
16.3.1	Time-base unit . . . . .	398
16.3.2	Counter modes . . . . .	400
16.3.3	Clock selection . . . . .	410
16.3.4	Capture/Compare channels . . . . .	414
16.3.5	Input capture mode . . . . .	416
16.3.6	PWM input mode . . . . .	417
16.3.7	Forced output mode . . . . .	418
16.3.8	Output compare mode . . . . .	418
16.3.9	PWM mode . . . . .	419
16.3.10	Asymmetric PWM mode . . . . .	423
16.3.11	Combined PWM mode . . . . .	423
16.3.12	Clearing the OCxREF signal on an external event . . . . .	424
16.3.13	One-pulse mode . . . . .	426
16.3.14	Retriggerable one pulse mode . . . . .	427
16.3.15	Encoder interface mode . . . . .	428
16.3.16	UIF bit remapping . . . . .	430
16.3.17	Timer input XOR function . . . . .	430
16.3.18	Timers and external trigger synchronization . . . . .	430
16.3.19	Timer synchronization . . . . .	434

16.3.20	DMA burst mode . . . . .	439
16.3.21	Debug mode . . . . .	440
16.4	TIM3 registers . . . . .	441
16.4.1	TIM3 control register 1 (TIM3_CR1) . . . . .	441
16.4.2	TIM3 control register 2 (TIM3_CR2) . . . . .	442
16.4.3	TIM3 slave mode control register (TIM3_SMCR) . . . . .	444
16.4.4	TIM3 DMA/Interrupt enable register (TIM3_DIER) . . . . .	447
16.4.5	TIM3 status register (TIM3_SR) . . . . .	448
16.4.6	TIM3 event generation register (TIM3_EGR) . . . . .	450
16.4.7	TIM3 capture/compare mode register 1 (TIM3_CCMR1) . . . . .	451
16.4.8	TIM3 capture/compare mode register 1 [alternate] (TIM3_CCMR1) . . . . .	453
16.4.9	TIM3 capture/compare mode register 2 (TIM3_CCMR2) . . . . .	455
16.4.10	TIM3 capture/compare mode register 2 [alternate] (TIM3_CCMR2) . . . . .	456
16.4.11	TIM3 capture/compare enable register (TIM3_CCER) . . . . .	457
16.4.12	TIM3 counter (TIM3_CNT) . . . . .	458
16.4.13	TIM3 counter [alternate] (TIM3_CNT) . . . . .	459
16.4.14	TIM3 prescaler (TIM3_PSC) . . . . .	459
16.4.15	TIM3 auto-reload register (TIM3_ARR) . . . . .	460
16.4.16	TIM3 capture/compare register 1 (TIM3_CCR1) . . . . .	460
16.4.17	TIM3 capture/compare register 2 (TIM3_CCR2) . . . . .	461
16.4.18	TIM3 capture/compare register 3 (TIM3_CCR3) . . . . .	461
16.4.19	TIM3 capture/compare register 4 (TIM3_CCR4) . . . . .	462
16.4.20	TIM3 DMA control register (TIM3_DCR) . . . . .	463
16.4.21	TIM3 DMA address for full transfer (TIM3_DMAR) . . . . .	463
16.4.22	TIM3 alternate function option register 1 (TIM3_AF1) . . . . .	464
16.4.23	TIM3 timer input selection register (TIM3_TISEL) . . . . .	464
16.4.24	TIMx register map . . . . .	465
17	<b>General-purpose timers (TIM14) . . . . .</b>	<b>467</b>
17.1	TIM14 introduction . . . . .	467
17.2	TIM14 main features . . . . .	467
17.2.1	TIM14 main features . . . . .	467
17.3	TIM14 functional description . . . . .	469
17.3.1	Time-base unit . . . . .	469
17.3.2	Counter modes . . . . .	471
17.3.3	Clock selection . . . . .	474

---

17.3.4	Capture/compare channels . . . . .	475
17.3.5	Input capture mode . . . . .	476
17.3.6	Forced output mode . . . . .	477
17.3.7	Output compare mode . . . . .	478
17.3.8	PWM mode . . . . .	479
17.3.9	One-pulse mode . . . . .	480
17.3.10	UIF bit remapping . . . . .	480
17.3.11	Using timer output as trigger for other timers (TIM14) . . . . .	481
17.3.12	Debug mode . . . . .	481
17.4	TIM14 registers . . . . .	482
17.4.1	TIM14 control register 1 (TIM14_CR1) . . . . .	482
17.4.2	TIM14 Interrupt enable register (TIM14_DIER) . . . . .	483
17.4.3	TIM14 status register (TIM14_SR) . . . . .	483
17.4.4	TIM14 event generation register (TIM14_EGR) . . . . .	484
17.4.5	TIM14 capture/compare mode register 1 (TIM14_CCMR1) . . . . .	485
17.4.6	TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1) . . . . .	486
17.4.7	TIM14 capture/compare enable register (TIM14_CCER) . . . . .	488
17.4.8	TIM14 counter (TIM14_CNT) . . . . .	489
17.4.9	TIM14 prescaler (TIM14_PSC) . . . . .	490
17.4.10	TIM14 auto-reload register (TIM14_ARR) . . . . .	490
17.4.11	TIM14 capture/compare register 1 (TIM14_CCR1) . . . . .	490
17.4.12	TIM14 timer input selection register (TIM14_TISEL) . . . . .	491
17.4.13	TIM14 register map . . . . .	491
<b>18</b>	<b>General-purpose timers (TIM16/TIM17) . . . . .</b>	<b>493</b>
18.1	TIM16/TIM17 introduction . . . . .	493
18.2	TIM16/TIM17 main features . . . . .	493
18.3	TIM16/TIM17 functional description . . . . .	495
18.3.1	Time-base unit . . . . .	495
18.3.2	Counter modes . . . . .	497
18.3.3	Repetition counter . . . . .	501
18.3.4	Clock selection . . . . .	502
18.3.5	Capture/compare channels . . . . .	504
18.3.6	Input capture mode . . . . .	506
18.3.7	Forced output mode . . . . .	508
18.3.8	Output compare mode . . . . .	508
18.3.9	PWM mode . . . . .	509

18.3.10	Complementary outputs and dead-time insertion .....	510
18.3.11	Using the break function .....	513
18.3.12	Bidirectional break inputs .....	517
18.3.13	6-step PWM generation .....	518
18.3.14	One-pulse mode .....	520
18.3.15	UIF bit remapping .....	521
18.3.16	Slave mode – combined reset + trigger mode .....	521
18.3.17	DMA burst mode .....	521
18.3.18	Using timer output as trigger for other timers (TIM16/TIM17) .....	522
18.3.19	Debug mode .....	523
18.4	TIM16/TIM17 registers .....	524
18.4.1	TIMx control register 1 (TIMx_CR1)(x = 16 to 17) .....	524
18.4.2	TIMx control register 2 (TIMx_CR2)(x = 16 to 17) .....	525
18.4.3	TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) .....	526
18.4.4	TIMx status register (TIMx_SR)(x = 16 to 17) .....	527
18.4.5	TIMx event generation register (TIMx_EGR)(x = 16 to 17) .....	528
18.4.6	TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16 to 17)	529
18.4.7	TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) .....	530
18.4.8	TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) .....	532
18.4.9	TIMx counter (TIMx_CNT)(x = 16 to 17) .....	534
18.4.10	TIMx prescaler (TIMx_PSC)(x = 16 to 17) .....	535
18.4.11	TIMx auto-reload register (TIMx_ARR)(x = 16 to 17) .....	535
18.4.12	TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) .....	536
18.4.13	TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) .....	536
18.4.14	TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) .....	537
18.4.15	TIMx DMA control register (TIMx_DCR)(x = 16 to 17) .....	540
18.4.16	TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) .....	540
18.4.17	TIM16 alternate function register 1 (TIM16_AF1) .....	541
18.4.18	TIM16 input selection register (TIM16_TISEL) .....	541
18.4.19	TIM17 alternate function register 1 (TIM17_AF1) .....	542
18.4.20	TIM17 input selection register (TIM17_TISEL) .....	542
18.4.21	TIM16/TIM17 register map .....	543
<b>19</b>	<b>Infrared interface (IRTIM)</b> .....	<b>545</b>
<b>20</b>	<b>Independent watchdog (IWDG)</b> .....	<b>546</b>

---

20.1	Introduction	546
20.2	IWDG main features	546
20.3	IWDG functional description	546
20.3.1	IWDG block diagram	546
20.3.2	Window option	547
20.3.3	Hardware watchdog	548
20.3.4	Register access protection	548
20.3.5	Debug mode	548
20.4	IWDG registers	549
20.4.1	IWDG key register (IWDG_KR)	549
20.4.2	IWDG prescaler register (IWDG_PR)	550
20.4.3	IWDG reload register (IWDG_RLR)	551
20.4.4	IWDG status register (IWDG_SR)	552
20.4.5	IWDG window register (IWDG_WINR)	553
20.4.6	IWDG register map	554
<b>21</b>	<b>System window watchdog (WWDG)</b>	<b>555</b>
21.1	Introduction	555
21.2	WWDG main features	555
21.3	WWDG functional description	555
21.3.1	WWDG block diagram	556
21.3.2	Enabling the watchdog	556
21.3.3	Controlling the down-counter	556
21.3.4	How to program the watchdog timeout	556
21.3.5	Debug mode	558
21.4	WWDG interrupts	558
21.5	WWDG registers	558
21.5.1	WWDG control register (WWDG_CR)	558
21.5.2	WWDG configuration register (WWDG_CFR)	559
21.5.3	WWDG status register (WWDG_SR)	560
21.5.4	WWDG register map	560
<b>22</b>	<b>Real-time clock (RTC)</b>	<b>561</b>
22.1	Introduction	561
22.2	RTC main features	561
22.3	RTC functional description	562

22.3.1	RTC block diagram .....	562
22.3.2	RTC pins and internal signals .....	563
22.3.3	GPIO controlled by the RTC .....	563
22.3.4	Clock and prescalers .....	565
22.3.5	Real-time clock and calendar .....	566
22.3.6	Programmable alarms .....	566
22.3.7	RTC initialization and configuration .....	566
22.3.8	Reading the calendar .....	568
22.3.9	Resetting the RTC .....	569
22.3.10	RTC synchronization .....	569
22.3.11	RTC reference clock detection .....	570
22.3.12	RTC smooth digital calibration .....	570
22.3.13	Timestamp function .....	572
22.3.14	Calibration clock output .....	573
22.3.15	Alarm output .....	573
22.4	RTC low-power modes .....	574
22.5	RTC interrupts .....	574
22.6	RTC registers .....	574
22.6.1	RTC time register (RTC_TR) .....	575
22.6.2	RTC date register (RTC_DR) .....	576
22.6.3	RTC sub second register (RTC_SSR) .....	577
22.6.4	RTC initialization control and status register (RTC_ICSR) .....	577
22.6.5	RTC prescaler register (RTC_PRER) .....	579
22.6.6	RTC control register (RTC_CR) .....	579
22.6.7	RTC write protection register (RTC_WPR) .....	582
22.6.8	RTC calibration register (RTC_CALR) .....	582
22.6.9	RTC shift control register (RTC_SHIFTR) .....	583
22.6.10	RTC timestamp time register (RTC_TSTR) .....	584
22.6.11	RTC timestamp date register (RTC_TSDDR) .....	585
22.6.12	RTC timestamp sub second register (RTC_TSSSR) .....	585
22.6.13	RTC alarm A register (RTC_ALRMAR) .....	586
22.6.14	RTC alarm A sub second register (RTC_ALRMASSR) .....	587
22.6.15	RTC status register (RTC_SR) .....	587
22.6.16	RTC masked interrupt status register (RTC_MISR) .....	588
22.6.17	RTC status clear register (RTC_SCR) .....	589
22.6.18	RTC register map .....	590

---

<b>23</b>	<b>Inter-integrated circuit (I2C) interface</b>	<b>592</b>
23.1	Introduction	592
23.2	I2C main features	592
23.3	I2C implementation	593
23.4	I2C functional description	593
23.4.1	I2C block diagram	594
23.4.2	I2C pins and internal signals	595
23.4.3	I2C clock requirements	595
23.4.4	Mode selection	595
23.4.5	I2C initialization	596
23.4.6	Software reset	600
23.4.7	Data transfer	601
23.4.8	I2C slave mode	603
23.4.9	I2C master mode	612
23.4.10	I2C_TIMINGR register configuration examples	623
23.4.11	SMBus specific features	625
23.4.12	SMBus initialization	628
23.4.13	SMBus: I2C_TIMEOUTR register configuration examples	630
23.4.14	SMBus slave mode	630
23.4.15	Wake-up from Stop mode on address match	637
23.4.16	Error conditions	638
23.4.17	DMA requests	640
23.4.18	Debug mode	640
23.5	I2C low-power modes	641
23.6	I2C interrupts	642
23.7	I2C registers	643
23.7.1	I2C control register 1 (I2C_CR1)	643
23.7.2	I2C control register 2 (I2C_CR2)	646
23.7.3	I2C own address 1 register (I2C_OAR1)	648
23.7.4	I2C own address 2 register (I2C_OAR2)	648
23.7.5	I2C timing register (I2C_TIMINGR)	649
23.7.6	I2C timeout register (I2C_TIMEOUTR)	650
23.7.7	I2C interrupt and status register (I2C_ISR)	651
23.7.8	I2C interrupt clear register (I2C_ICR)	654
23.7.9	I2C PEC register (I2C_PECR)	655
23.7.10	I2C receive data register (I2C_RXDR)	655

23.7.11	I2C transmit data register (I2C_TXDR) . . . . .	656
23.7.12	I2C register map . . . . .	656
<b>24</b>	<b>Universal synchronous receiver transmitter (USART) . . . . .</b>	<b>658</b>
24.1	USART introduction . . . . .	658
24.2	USART main features . . . . .	659
24.3	USART extended features . . . . .	660
24.4	USART implementation . . . . .	660
24.5	USART functional description . . . . .	662
24.5.1	USART block diagram . . . . .	662
24.5.2	USART signals . . . . .	663
24.5.3	USART character description . . . . .	664
24.5.4	USART FIFOs and thresholds . . . . .	666
24.5.5	USART transmitter . . . . .	666
24.5.6	USART receiver . . . . .	670
24.5.7	USART baud rate generation . . . . .	677
24.5.8	Tolerance of the USART receiver to clock deviation . . . . .	678
24.5.9	USART auto baud rate detection . . . . .	680
24.5.10	USART multiprocessor communication . . . . .	682
24.5.11	USART Modbus communication . . . . .	684
24.5.12	USART parity control . . . . .	685
24.5.13	USART LIN (local interconnection network) mode . . . . .	686
24.5.14	USART synchronous mode . . . . .	688
24.5.15	USART single-wire Half-duplex communication . . . . .	692
24.5.16	USART receiver timeout . . . . .	692
24.5.17	USART Smartcard mode . . . . .	693
24.5.18	USART IrDA SIR ENDEC block . . . . .	697
24.5.19	Continuous communication using USART and DMA . . . . .	700
24.5.20	RS232 Hardware flow control and RS485 Driver Enable . . . . .	702
24.5.21	USART low-power management . . . . .	705
24.6	USART in low-power modes . . . . .	708
24.7	USART interrupts . . . . .	709
24.8	USART registers . . . . .	710
24.8.1	USART control register 1 (USART_CR1) . . . . .	710
24.8.2	USART control register 1 [alternate] (USART_CR1) . . . . .	714
24.8.3	USART control register 2 (USART_CR2) . . . . .	717

---

24.8.4	USART control register 3 (USART_CR3) . . . . .	721
24.8.5	USART baud rate register (USART_BRR) . . . . .	726
24.8.6	USART guard time and prescaler register (USART_GTPR) . . . . .	726
24.8.7	USART receiver timeout register (USART_RTOR) . . . . .	727
24.8.8	USART request register (USART_RQR) . . . . .	728
24.8.9	USART interrupt and status register (USART_ISR) . . . . .	729
24.8.10	USART interrupt and status register [alternate] (USART_ISR) . . . . .	735
24.8.11	USART interrupt flag clear register (USART_ICR) . . . . .	740
24.8.12	USART receive data register (USART_RDR) . . . . .	742
24.8.13	USART transmit data register (USART_TDR) . . . . .	742
24.8.14	USART prescaler register (USART_PRESC) . . . . .	743
24.8.15	USART register map . . . . .	744
<b>25</b>	<b>Serial peripheral interface / integrated interchip sound (SPI/I2S) . . . . .</b>	<b>746</b>
25.1	Introduction . . . . .	746
25.2	SPI main features . . . . .	746
25.3	I2S main features . . . . .	747
25.4	SPI/I2S implementation . . . . .	747
25.5	SPI functional description . . . . .	748
25.5.1	General description . . . . .	748
25.5.2	Communications between one master and one slave . . . . .	749
25.5.3	Standard multislave communication . . . . .	751
25.5.4	Multimaster communication . . . . .	752
25.5.5	Slave select (NSS) pin management . . . . .	753
25.5.6	Communication formats . . . . .	754
25.5.7	Configuration of SPI . . . . .	756
25.5.8	Procedure for enabling SPI . . . . .	757
25.5.9	Data transmission and reception procedures . . . . .	757
25.5.10	SPI status flags . . . . .	767
25.5.11	SPI error flags . . . . .	768
25.5.12	NSS pulse mode . . . . .	769
25.5.13	TI mode . . . . .	769
25.5.14	CRC calculation . . . . .	770
25.6	SPI interrupts . . . . .	772
25.7	I2S functional description . . . . .	773
25.7.1	I2S general description . . . . .	773

25.7.2	Supported audio protocols . . . . .	774
25.7.3	Start-up description . . . . .	781
25.7.4	Clock generator . . . . .	783
25.7.5	I <sup>2</sup> S master mode . . . . .	786
25.7.6	I <sup>2</sup> S slave mode . . . . .	787
25.7.7	I2S status flags . . . . .	789
25.7.8	I2S error flags . . . . .	790
25.7.9	DMA features . . . . .	791
25.8	I2S interrupts . . . . .	791
25.9	SPI and I2S registers . . . . .	792
25.9.1	SPI control register 1 (SPIx_CR1) . . . . .	792
25.9.2	SPI control register 2 (SPIx_CR2) . . . . .	794
25.9.3	SPI status register (SPIx_SR) . . . . .	796
25.9.4	SPI data register (SPIx_DR) . . . . .	798
25.9.5	SPI CRC polynomial register (SPIx_CRCPR) . . . . .	798
25.9.6	SPI Rx CRC register (SPIx_RXCRCR) . . . . .	798
25.9.7	SPI Tx CRC register (SPIx_TXCRCR) . . . . .	799
25.9.8	SPIx_I2S configuration register (SPIx_I2SCFGR) . . . . .	799
25.9.9	SPIx_I2S prescaler register (SPIx_I2SPR) . . . . .	801
25.9.10	SPI/I2S register map . . . . .	803
<b>26</b>	<b>Debug support (DBG) . . . . .</b>	<b>804</b>
26.1	Overview . . . . .	804
26.2	Reference Arm documentation . . . . .	805
26.3	Pinout and debug port pins . . . . .	805
26.3.1	SWD port pins . . . . .	805
26.3.2	SW-DP pin assignment . . . . .	805
26.3.3	Internal pull-up & pull-down on SWD pins . . . . .	806
26.4	ID codes and locking mechanism . . . . .	806
26.5	SWD port . . . . .	806
26.5.1	SWD protocol introduction . . . . .	806
26.5.2	SWD protocol sequence . . . . .	806
26.5.3	SW-DP state machine (reset, idle states, ID code) . . . . .	807
26.5.4	DP and AP read/write accesses . . . . .	808
26.5.5	SW-DP registers . . . . .	808
26.5.6	SW-AP registers . . . . .	809

---

26.6	Core debug .....	810
26.7	BPU (break point unit) .....	810
26.7.1	BPU functionality .....	811
26.8	DWT (data watchpoint) .....	811
26.8.1	DWT functionality .....	811
26.8.2	DWT Program counter sample register .....	811
26.9	MCU debug component (DBG) .....	811
26.9.1	Debug support for low-power modes .....	811
26.9.2	Debug support for timers, watchdog, and I2C .....	812
26.10	DBG registers .....	812
26.10.1	DBG device ID code register (DBG_IDCODE) .....	812
26.10.2	DBG configuration register (DBG_CR) .....	813
26.10.3	DBG APB freeze register 1 (DBG_APB_FZ1) .....	813
26.10.4	DBG APB freeze register 2 (DBG_APB_FZ2) .....	814
26.10.5	DBG register map .....	815
<b>27</b>	<b>Device electronic signature .....</b>	<b>817</b>
27.1	Unique device ID register (96 bits) .....	817
27.2	Flash memory size data register .....	818
27.3	Package data register .....	818
<b>28</b>	<b>Important security notice .....</b>	<b>820</b>
<b>29</b>	<b>Revision history .....</b>	<b>821</b>

## List of tables

Table 1.	STM32C0x1 boundary addresses . . . . .	40
Table 2.	STM32C0x1 peripheral register boundary addresses . . . . .	41
Table 3.	SRAM size . . . . .	43
Table 4.	Boot modes. . . . .	44
Table 5.	Flash memory organization . . . . .	48
Table 6.	LATENCY[2:0] setting as function of HCLK frequency. . . . .	49
Table 7.	Page erase overview . . . . .	51
Table 8.	Mass erase overview . . . . .	52
Table 9.	Option byte format . . . . .	56
Table 10.	Organization of option bytes . . . . .	56
Table 11.	Flash memory read protection status . . . . .	59
Table 13.	Access status versus protection level and execution modes . . . . .	61
Table 16.	Securable memory erase at RDP Level 1 to Level 0 change . . . . .	64
Table 17.	FLASH interrupt requests . . . . .	65
Table 18.	FLASH register map and reset values . . . . .	79
Table 19.	Device resources enabled in different operating modes. . . . .	85
Table 20.	Low-power mode entry overview . . . . .	88
Table 21.	Low-power mode exit overview . . . . .	89
Table 22.	PWR register map and reset values . . . . .	101
Table 23.	RCC register map and reset values . . . . .	142
Table 24.	Port bit configuration table . . . . .	146
Table 25.	GPIO register map and reset values . . . . .	160
Table 26.	SYSCFG register map and reset values. . . . .	174
Table 27.	Interconnect matrix . . . . .	177
Table 28.	DMA implementation . . . . .	183
Table 29.	DMA internal input/output signals . . . . .	184
Table 30.	Programmable data width and endian behavior (when PINC = MINC = 1) . . . . .	189
Table 31.	DMA interrupt requests. . . . .	191
Table 32.	DMA register map and reset values . . . . .	197
Table 33.	DMAMUX instantiation . . . . .	200
Table 34.	DMAMUX: assignment of multiplexer inputs to resources . . . . .	200
Table 35.	DMAMUX: assignment of trigger inputs to resources. . . . .	201
Table 36.	DMAMUX: assignment of synchronization inputs to resources . . . . .	201
Table 37.	DMAMUX signals . . . . .	204
Table 38.	DMAMUX interrupts . . . . .	208
Table 39.	DMAMUX register map and reset values . . . . .	213
Table 40.	Vector table. . . . .	214
Table 41.	EXTI signal overview . . . . .	218
Table 42.	EVG pin overview . . . . .	218
Table 43.	EXTI event input configurations and register control . . . . .	219
Table 44.	EXTI line connections . . . . .	222
Table 45.	Masking functionality . . . . .	222
Table 46.	EXTI register map sections. . . . .	223
Table 47.	EXTI controller register map and reset values . . . . .	229
Table 48.	CRC internal input/output signals . . . . .	232
Table 49.	CRC register map and reset values . . . . .	237
Table 50.	ADC main features . . . . .	240
Table 51.	ADC input/output pins. . . . .	241

Table 52.	ADC internal input/output signals . . . . .	242
Table 53.	External triggers . . . . .	242
Table 54.	Latency between trigger and start of conversion . . . . .	247
Table 55.	Configuring the trigger polarity . . . . .	254
Table 56.	tSAR timings depending on resolution . . . . .	256
Table 57.	Analog watchdog comparison. . . . .	265
Table 58.	Analog watchdog 1 channel selection . . . . .	266
Table 59.	Maximum output results vs N and M. Grayed values indicates truncation . . . . .	270
Table 60.	ADC interrupts . . . . .	275
Table 61.	ADC register map and reset values . . . . .	295
Table 62.	Behavior of timer outputs versus BRK/BRK2 inputs . . . . .	339
Table 63.	Break protection disarming conditions . . . . .	341
Table 64.	Counting direction versus encoder signals . . . . .	347
Table 65.	TIM1 internal trigger connection . . . . .	364
Table 66.	Output control bits for complementary OCx and OCxN channels with break feature . . . . .	378
Table 67.	TIM1 register map and reset values . . . . .	393
Table 68.	Counting direction versus encoder signals . . . . .	429
Table 69.	TIM3 internal trigger connection . . . . .	447
Table 70.	Output control bit for standard OCx channels . . . . .	458
Table 71.	TIM3 register map and reset values . . . . .	465
Table 72.	Output control bit for standard OCx channels . . . . .	489
Table 73.	TIM14 register map and reset values . . . . .	491
Table 74.	Break protection disarming conditions . . . . .	517
Table 75.	Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) . . . . .	534
Table 76.	TIM16/TIM17 register map and reset values . . . . .	543
Table 77.	IWDG register map and reset values . . . . .	554
Table 78.	WWDG register map and reset values . . . . .	560
Table 79.	RTC input/output pins . . . . .	563
Table 80.	RTC internal input/output signals . . . . .	563
Table 81.	Pin configuration . . . . .	564
Table 82.	RTC_OUT mapping . . . . .	565
Table 83.	Effect of low-power modes on RTC . . . . .	574
Table 84.	RTC pins functionality over modes . . . . .	574
Table 85.	Interrupt requests . . . . .	574
Table 86.	RTC register map and reset values . . . . .	590
Table 87.	STM32C0x1 I2C implementation . . . . .	593
Table 88.	I2C input/output pins . . . . .	595
Table 89.	I2C internal input/output signals . . . . .	595
Table 90.	Comparison of analog vs. digital filters . . . . .	597
Table 91.	I2C-SMBus specification data setup and hold times . . . . .	599
Table 92.	I2C configuration . . . . .	603
Table 93.	I2C-SMBus specification clock timings . . . . .	614
Table 94.	Examples of timing settings for fI2CCLK = 8 MHz . . . . .	624
Table 95.	Examples of timing settings for fI2CCLK = 16 MHz . . . . .	624
Table 96.	Examples of timing settings for fI2CCLK = 48 MHz . . . . .	625
Table 97.	SMBus timeout specifications . . . . .	627
Table 98.	SMBus with PEC configuration . . . . .	628
Table 99.	Examples of TIMEOUTA settings (max t <sub>TIMEOUT</sub> = 25 ms) . . . . .	630
Table 100.	Examples of TIMEOUTB settings . . . . .	630
Table 101.	Examples of TIMEOUTA settings (max t <sub>IDLE</sub> = 50 µs) . . . . .	630
Table 102.	Effect of low-power modes on the I2C . . . . .	641

Table 103. I2C Interrupt requests . . . . .	642
Table 104. I2C register map and reset values . . . . .	656
Table 105. STM32C0x1 features . . . . .	660
Table 106. USART features . . . . .	661
Table 107. Noise detection from sampled data . . . . .	676
Table 108. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . . .	679
Table 109. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . . .	680
Table 110. USART frame formats . . . . .	685
Table 111. Effect of low-power modes on the USART . . . . .	708
Table 112. USART interrupt requests . . . . .	709
Table 113. USART register map and reset values . . . . .	744
Table 114. STM32C0x1 SPI and SPI/I2S implementation . . . . .	747
Table 115. SPI interrupt requests . . . . .	772
Table 116. Audio-frequency precision using 48 MHz clock derived from HSE . . . . .	785
Table 117. I2S interrupt requests . . . . .	791
Table 118. SPI/I2S register map and reset values . . . . .	803
Table 119. SW debug port pins . . . . .	805
Table 120. Packet request (8-bits) . . . . .	807
Table 121. ACK response (3 bits) . . . . .	807
Table 122. DATA transfer (33 bits) . . . . .	807
Table 123. SW-DP registers . . . . .	808
Table 124. 32-bit debug port registers addressed through the shifted value A[3:2] . . . . .	809
Table 125. Core debug registers . . . . .	810
Table 126. DEV_ID and REV_ID field values . . . . .	812
Table 127. DBG register map and reset values . . . . .	816
Table 128. Document revision history . . . . .	821

## List of figures

Figure 1.	System architecture . . . . .	37
Figure 2.	Memory map . . . . .	40
Figure 3.	Changing read protection (RDP) level . . . . .	61
Figure 4.	Example of disabling core debug access . . . . .	65
Figure 5.	Power supply overview . . . . .	81
Figure 6.	POR, PDR, and BOR thresholds . . . . .	83
Figure 7.	Low-power mode transit diagram . . . . .	84
Figure 8.	Simplified diagram of the reset circuit . . . . .	104
Figure 9.	Clock tree . . . . .	108
Figure 10.	HSE/ LSE clock sources . . . . .	109
Figure 11.	Frequency measurement with TIM14 in capture mode . . . . .	114
Figure 12.	Frequency measurement with TIM16 in capture mode . . . . .	115
Figure 13.	Frequency measurement with TIM17 in capture mode . . . . .	115
Figure 14.	Basic structure of an I/O port bit . . . . .	146
Figure 15.	Input floating/pull up/pull down configurations . . . . .	150
Figure 16.	Output configuration . . . . .	151
Figure 17.	Alternate function configuration- . . . . .	152
Figure 18.	High impedance-analog configuration . . . . .	152
Figure 19.	DMA block diagram . . . . .	183
Figure 20.	DMAMUX block diagram . . . . .	203
Figure 21.	Synchronization mode of the DMAMUX request line multiplexer channel . . . . .	206
Figure 22.	Event generation of the DMA request line multiplexer channel . . . . .	206
Figure 23.	EXTI block diagram . . . . .	218
Figure 24.	Configurable event trigger logic CPU wakeup . . . . .	220
Figure 25.	Direct event trigger logic CPU wakeup . . . . .	221
Figure 26.	EXTI GPIO multiplexer . . . . .	221
Figure 27.	CRC calculation unit block diagram . . . . .	232
Figure 28.	ADC block diagram . . . . .	241
Figure 29.	ADC calibration . . . . .	244
Figure 30.	Calibration factor forcing . . . . .	244
Figure 31.	Enabling/disabling the ADC . . . . .	245
Figure 32.	ADC clock scheme . . . . .	246
Figure 33.	ADC connectivity . . . . .	248
Figure 34.	Analog to digital conversion time . . . . .	253
Figure 35.	ADC conversion timings . . . . .	253
Figure 36.	Stopping an ongoing conversion . . . . .	254
Figure 37.	Single conversions of a sequence, software trigger . . . . .	257
Figure 38.	Continuous conversion of a sequence, software trigger . . . . .	257
Figure 39.	Single conversions of a sequence, hardware trigger . . . . .	258
Figure 40.	Continuous conversions of a sequence, hardware trigger . . . . .	258
Figure 41.	Data alignment and resolution (oversampling disabled: OVSE = 0) . . . . .	259
Figure 42.	Example of overrun (OVR) . . . . .	260
Figure 43.	Wait mode conversion (continuous mode, software trigger) . . . . .	263
Figure 44.	Behavior with WAIT = 0, AUTOFF = 1 . . . . .	264
Figure 45.	Behavior with WAIT = 1, AUTOFF = 1 . . . . .	264
Figure 46.	Analog watchdog guarded area . . . . .	265
Figure 47.	ADC_AWDx_OUT signal generation . . . . .	267
Figure 48.	ADC_AWDx_OUT signal generation (AWDx flag not cleared by software) . . . . .	267

---

Figure 49.	ADC_AWDx_OUT signal generation (on a single channel) . . . . .	268
Figure 50.	Analog watchdog threshold update . . . . .	268
Figure 51.	20-bit to 16-bit result truncation . . . . .	269
Figure 52.	Numerical example with 5-bits shift and rounding . . . . .	270
Figure 53.	Triggered oversampling mode (TOVS bit = 1) . . . . .	272
Figure 54.	Temperature sensor and VREFINT channel block diagram . . . . .	273
Figure 55.	Advanced-control timer block diagram . . . . .	299
Figure 56.	Counter timing diagram with prescaler division change from 1 to 2 . . . . .	301
Figure 57.	Counter timing diagram with prescaler division change from 1 to 4 . . . . .	301
Figure 58.	Counter timing diagram, internal clock divided by 1 . . . . .	303
Figure 59.	Counter timing diagram, internal clock divided by 2 . . . . .	303
Figure 60.	Counter timing diagram, internal clock divided by 4 . . . . .	304
Figure 61.	Counter timing diagram, internal clock divided by N . . . . .	304
Figure 62.	Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . . .	305
Figure 63.	Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . . .	305
Figure 64.	Counter timing diagram, internal clock divided by 1 . . . . .	307
Figure 65.	Counter timing diagram, internal clock divided by 2 . . . . .	307
Figure 66.	Counter timing diagram, internal clock divided by 4 . . . . .	308
Figure 67.	Counter timing diagram, internal clock divided by N . . . . .	308
Figure 68.	Counter timing diagram, update event when repetition counter is not used . . . . .	309
Figure 69.	Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . . .	310
Figure 70.	Counter timing diagram, internal clock divided by 2 . . . . .	311
Figure 71.	Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . .	311
Figure 72.	Counter timing diagram, internal clock divided by N . . . . .	312
Figure 73.	Counter timing diagram, update event with ARPE=1 (counter underflow) . . . . .	312
Figure 74.	Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . . .	313
Figure 75.	Update rate examples depending on mode and TIMx_RCR register settings . . . . .	314
Figure 76.	External trigger input block . . . . .	315
Figure 77.	TIM1 ETR input circuitry . . . . .	315
Figure 78.	Control circuit in normal mode, internal clock divided by 1 . . . . .	316
Figure 79.	TI2 external clock connection example . . . . .	317
Figure 80.	Control circuit in external clock mode 1 . . . . .	318
Figure 81.	External trigger input block . . . . .	318
Figure 82.	Control circuit in external clock mode 2 . . . . .	319
Figure 83.	Capture/compare channel (example: channel 1 input stage) . . . . .	320
Figure 84.	Capture/compare channel 1 main circuit . . . . .	320
Figure 85.	Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) . . . . .	321
Figure 86.	Output stage of capture/compare channel (channel 4) . . . . .	321
Figure 87.	Output stage of capture/compare channel (channel 5, idem ch. 6) . . . . .	322
Figure 88.	PWM input mode timing . . . . .	324
Figure 89.	Output compare mode, toggle on OC1 . . . . .	326
Figure 90.	Edge-aligned PWM waveforms (ARR=8) . . . . .	327
Figure 91.	Center-aligned PWM waveforms (ARR=8) . . . . .	328
Figure 92.	Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . .	330
Figure 93.	Combined PWM mode on channel 1 and 3 . . . . .	331
Figure 94.	3-phase combined PWM signals with multiple trigger pulses per period . . . . .	332
Figure 95.	Complementary output with dead-time insertion . . . . .	333
Figure 96.	Dead-time waveforms with delay greater than the negative pulse . . . . .	333
Figure 97.	Dead-time waveforms with delay greater than the positive pulse . . . . .	334
Figure 98.	Break and Break2 circuitry overview . . . . .	336
Figure 99.	Various output behavior in response to a break event on BRK (OSSI = 1) . . . . .	338
Figure 100.	PWM output state following BRK and BRK2 pins assertion (OSSI=1) . . . . .	339

---

Figure 101. PWM output state following BRK assertion (OSSI=0) . . . . .	340
Figure 102. Output redirection (BRK2 request not represented) . . . . .	341
Figure 103. Clearing TIMx OCxREF . . . . .	342
Figure 104. 6-step generation, COM example (OSSR=1) . . . . .	343
Figure 105. Example of one pulse mode. . . . .	344
Figure 106. Retriggerable one pulse mode . . . . .	346
Figure 107. Example of counter operation in encoder interface mode. . . . .	347
Figure 108. Example of encoder interface mode with TI1FP1 polarity inverted. . . . .	348
Figure 109. Measuring time interval between edges on 3 signals . . . . .	349
Figure 110. Example of Hall sensor interface . . . . .	351
Figure 111. Control circuit in reset mode . . . . .	352
Figure 112. Control circuit in Gated mode . . . . .	353
Figure 113. Control circuit in trigger mode . . . . .	354
Figure 114. Control circuit in external clock mode 2 + trigger mode . . . . .	355
Figure 115. General-purpose timer block diagram . . . . .	397
Figure 116. Counter timing diagram with prescaler division change from 1 to 2 . . . . .	399
Figure 117. Counter timing diagram with prescaler division change from 1 to 4 . . . . .	399
Figure 118. Counter timing diagram, internal clock divided by 1 . . . . .	400
Figure 119. Counter timing diagram, internal clock divided by 2 . . . . .	401
Figure 120. Counter timing diagram, internal clock divided by 4 . . . . .	401
Figure 121. Counter timing diagram, internal clock divided by N . . . . .	402
Figure 122. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded) . . . . .	402
Figure 123. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded) . . . . .	403
Figure 124. Counter timing diagram, internal clock divided by 1 . . . . .	404
Figure 125. Counter timing diagram, internal clock divided by 2 . . . . .	404
Figure 126. Counter timing diagram, internal clock divided by 4 . . . . .	405
Figure 127. Counter timing diagram, internal clock divided by N . . . . .	405
Figure 128. Counter timing diagram, Update event . . . . .	406
Figure 129. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . . .	407
Figure 130. Counter timing diagram, internal clock divided by 2 . . . . .	408
Figure 131. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . .	408
Figure 132. Counter timing diagram, internal clock divided by N . . . . .	409
Figure 133. Counter timing diagram, Update event with ARPE=1 (counter underflow) . . . . .	409
Figure 134. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . . .	410
Figure 135. Control circuit in normal mode, internal clock divided by 1 . . . . .	411
Figure 136. TI2 external clock connection example. . . . .	411
Figure 137. Control circuit in external clock mode 1 . . . . .	412
Figure 138. External trigger input block . . . . .	413
Figure 139. Control circuit in external clock mode 2 . . . . .	414
Figure 140. Capture/Compare channel (example: channel 1 input stage) . . . . .	414
Figure 141. Capture/Compare channel 1 main circuit . . . . .	415
Figure 142. Output stage of Capture/Compare channel (channel 1) . . . . .	415
Figure 143. PWM input mode timing . . . . .	417
Figure 144. Output compare mode, toggle on OC1 . . . . .	419
Figure 145. Edge-aligned PWM waveforms (ARR=8) . . . . .	420
Figure 146. Center-aligned PWM waveforms (ARR=8) . . . . .	422
Figure 147. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . .	423
Figure 148. Combined PWM mode on channels 1 and 3 . . . . .	424
Figure 149. Clearing TIMx OCxREF . . . . .	425
Figure 150. Example of one-pulse mode. . . . .	426
Figure 151. Retriggerable one-pulse mode . . . . .	428
Figure 152. Example of counter operation in encoder interface mode . . . . .	429

Figure 153. Example of encoder interface mode with TI1FP1 polarity inverted .....	430
Figure 154. Control circuit in reset mode .....	431
Figure 155. Control circuit in gated mode .....	432
Figure 156. Control circuit in trigger mode .....	433
Figure 157. Control circuit in external clock mode 2 + trigger mode .....	434
Figure 158. Master/Slave timer example .....	434
Figure 159. Master/slave connection example with 1 channel only timers .....	435
Figure 160. Gating TIMz with OC1REF of TIMy .....	436
Figure 161. Gating TIMz with Enable of TIMy .....	437
Figure 162. Triggering TIMz with update of TIMy .....	437
Figure 163. Triggering TIMz with Enable of TIMy .....	438
Figure 164. Triggering TIMy and TIMz with TIMy TI1 input .....	439
Figure 165. General-purpose timer block diagram (TIM14) .....	468
Figure 166. Counter timing diagram with prescaler division change from 1 to 2 .....	470
Figure 167. Counter timing diagram with prescaler division change from 1 to 4 .....	470
Figure 168. Counter timing diagram, internal clock divided by 1 .....	471
Figure 169. Counter timing diagram, internal clock divided by 2 .....	472
Figure 170. Counter timing diagram, internal clock divided by 4 .....	472
Figure 171. Counter timing diagram, internal clock divided by N .....	473
Figure 172. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	473
Figure 173. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) .....	474
Figure 174. Control circuit in normal mode, internal clock divided by 1 .....	475
Figure 175. Capture/compare channel (example: channel 1 input stage) .....	475
Figure 176. Capture/compare channel 1 main circuit .....	476
Figure 177. Output stage of capture/compare channel (channel 1) .....	476
Figure 178. Output compare mode, toggle on OC1 .....	479
Figure 179. Edge-aligned PWM waveforms (ARR=8) .....	480
Figure 180. TIM16/TIM17 block diagram .....	494
Figure 181. Counter timing diagram with prescaler division change from 1 to 2 .....	496
Figure 182. Counter timing diagram with prescaler division change from 1 to 4 .....	496
Figure 183. Counter timing diagram, internal clock divided by 1 .....	498
Figure 184. Counter timing diagram, internal clock divided by 2 .....	498
Figure 185. Counter timing diagram, internal clock divided by 4 .....	499
Figure 186. Counter timing diagram, internal clock divided by N .....	499
Figure 187. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) .....	500
Figure 188. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) .....	500
Figure 189. Update rate examples depending on mode and TIMx_RCR register settings .....	502
Figure 190. Control circuit in normal mode, internal clock divided by 1 .....	503
Figure 191. TI2 external clock connection example .....	503
Figure 192. Control circuit in external clock mode 1 .....	504
Figure 193. Capture/compare channel (example: channel 1 input stage) .....	505
Figure 194. Capture/compare channel 1 main circuit .....	505
Figure 195. Output stage of capture/compare channel (channel 1) .....	506
Figure 196. Output compare mode, toggle on OC1 .....	509
Figure 197. Edge-aligned PWM waveforms (ARR=8) .....	510
Figure 198. Complementary output with dead-time insertion .....	511
Figure 199. Dead-time waveforms with delay greater than the negative pulse .....	512
Figure 200. Dead-time waveforms with delay greater than the positive pulse .....	512

---

Figure 201. Break circuitry overview . . . . .	514
Figure 202. Output behavior in response to a break . . . . .	516
Figure 203. Output redirection . . . . .	518
Figure 204. 6-step generation, COM example (OSSR=1) . . . . .	519
Figure 205. Example of one pulse mode . . . . .	520
Figure 206. IRTIM internal hardware connections with TIM16 and TIM17 . . . . .	545
Figure 207. Independent watchdog block diagram . . . . .	546
Figure 208. Watchdog block diagram . . . . .	556
Figure 209. Window watchdog timing diagram . . . . .	557
Figure 210. RTC block diagram . . . . .	562
Figure 211. I2C block diagram . . . . .	594
Figure 212. I2C bus protocol . . . . .	596
Figure 213. Setup and hold timings . . . . .	598
Figure 214. I2C initialization flow . . . . .	600
Figure 215. Data reception . . . . .	601
Figure 216. Data transmission . . . . .	602
Figure 217. Slave initialization flow . . . . .	605
Figure 218. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0 . . . . .	607
Figure 219. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1 . . . . .	608
Figure 220. Transfer bus diagrams for I2C slave transmitter (mandatory events only) . . . . .	609
Figure 221. Transfer sequence flow for slave receiver with NOSTRETCH = 0 . . . . .	610
Figure 222. Transfer sequence flow for slave receiver with NOSTRETCH = 1 . . . . .	611
Figure 223. Transfer bus diagrams for I2C slave receiver (mandatory events only) . . . . .	611
Figure 224. Master clock generation . . . . .	613
Figure 225. Master initialization flow . . . . .	615
Figure 226. 10-bit address read access with HEAD10R = 0 . . . . .	615
Figure 227. 10-bit address read access with HEAD10R = 1 . . . . .	616
Figure 228. Transfer sequence flow for I2C master transmitter for $N \leq 255$ bytes . . . . .	617
Figure 229. Transfer sequence flow for I2C master transmitter for $N > 255$ bytes . . . . .	618
Figure 230. Transfer bus diagrams for I2C master transmitter (mandatory events only) . . . . .	619
Figure 231. Transfer sequence flow for I2C master receiver for $N \leq 255$ bytes . . . . .	621
Figure 232. Transfer sequence flow for I2C master receiver for $N > 255$ bytes . . . . .	622
Figure 233. Transfer bus diagrams for I2C master receiver (mandatory events only) . . . . .	623
Figure 234. Timeout intervals for $t_{LOW:SEXT}$ , $t_{LOW:MEXT}$ . . . . .	627
Figure 235. Transfer sequence flow for SMBus slave transmitter N bytes + PEC . . . . .	631
Figure 236. Transfer bus diagrams for SMBus slave transmitter (SBC = 1) . . . . .	631
Figure 237. Transfer sequence flow for SMBus slave receiver N bytes + PEC . . . . .	633
Figure 238. Bus transfer diagrams for SMBus slave receiver (SBC = 1) . . . . .	634
Figure 239. Bus transfer diagrams for SMBus master transmitter . . . . .	635
Figure 240. Bus transfer diagrams for SMBus master receiver . . . . .	637
Figure 241. USART block diagram . . . . .	662
Figure 242. Word length programming . . . . .	665
Figure 243. Configurable stop bits . . . . .	667
Figure 244. TC/TXE behavior when transmitting . . . . .	670
Figure 245. Start bit detection when oversampling by 16 or 8 . . . . .	671
Figure 246. usart_ker_ck clock divider block diagram . . . . .	674
Figure 247. Data sampling when oversampling by 16 . . . . .	675
Figure 248. Data sampling when oversampling by 8 . . . . .	676
Figure 249. Mute mode using Idle line detection . . . . .	683

---

Figure 250. Mute mode using address mark detection . . . . .	684
Figure 251. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . . .	687
Figure 252. Break detection in LIN mode vs. Framing error detection. . . . .	688
Figure 253. USART example of synchronous master transmission. . . . .	689
Figure 254. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . . .	689
Figure 255. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . . .	690
Figure 256. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . . .	691
Figure 257. ISO 7816-3 asynchronous protocol . . . . .	693
Figure 258. Parity error detection using the 1.5 stop bits . . . . .	695
Figure 259. IrDA SIR ENDEC block diagram. . . . .	699
Figure 260. IrDA data modulation (3/16) - Normal mode . . . . .	699
Figure 261. Transmission using DMA . . . . .	701
Figure 262. Reception using DMA . . . . .	702
Figure 263. Hardware flow control between 2 USARTs . . . . .	702
Figure 264. RS232 RTS flow control . . . . .	703
Figure 265. RS232 CTS flow control . . . . .	704
Figure 266. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . . .	707
Figure 267. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . . .	707
Figure 268. SPI block diagram. . . . .	748
Figure 269. Full-duplex single master/ single slave application. . . . .	749
Figure 270. Half-duplex single master/ single slave application . . . . .	750
Figure 271. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode) . . . . .	751
Figure 272. Master and three independent slaves. . . . .	752
Figure 273. Multimaster application . . . . .	753
Figure 274. Hardware/software slave select management . . . . .	754
Figure 275. Data clock timing diagram . . . . .	755
Figure 276. Data alignment when data length is not equal to 8-bit or 16-bit . . . . .	756
Figure 277. Packing data in FIFO for transmission and reception. . . . .	760
Figure 278. Master full-duplex communication . . . . .	763
Figure 279. Slave full-duplex communication . . . . .	764
Figure 280. Master full-duplex communication with CRC . . . . .	765
Figure 281. Master full-duplex communication in packed mode . . . . .	766
Figure 282. NSSP pulse generation in Motorola SPI master mode. . . . .	769
Figure 283. TI mode transfer . . . . .	770
Figure 284. I <sup>2</sup> S block diagram . . . . .	773
Figure 285. I <sup>2</sup> S Philips protocol waveforms (16/32-bit full accuracy). . . . .	775
Figure 286. I <sup>2</sup> S Philips standard waveforms (24-bit frame) . . . . .	775
Figure 287. Transmitting 0x8EAA33 . . . . .	776
Figure 288. Receiving 0x8EAA33 . . . . .	776
Figure 289. I <sup>2</sup> S Philips standard (16-bit extended to 32-bit packet frame) . . . . .	776
Figure 290. Example of 16-bit data frame extended to 32-bit channel frame . . . . .	776
Figure 291. MSB Justified 16-bit or 32-bit full-accuracy length . . . . .	777
Figure 292. MSB justified 24-bit frame length . . . . .	777
Figure 293. MSB justified 16-bit extended to 32-bit packet frame . . . . .	778
Figure 294. LSB justified 16-bit or 32-bit full-accuracy . . . . .	778
Figure 295. LSB justified 24-bit frame length . . . . .	778
Figure 296. Operations required to transmit 0x3478AE . . . . .	779

---

Figure 297. Operations required to receive 0x3478AE . . . . .	779
Figure 298. LSB justified 16-bit extended to 32-bit packet frame . . . . .	779
Figure 299. Example of 16-bit data frame extended to 32-bit channel frame . . . . .	780
Figure 300. PCM standard waveforms (16-bit) . . . . .	780
Figure 301. PCM standard waveforms (16-bit extended to 32-bit packet frame) . . . . .	781
Figure 302. Start sequence in master mode . . . . .	782
Figure 303. Audio sampling frequency definition . . . . .	783
Figure 304. I <sup>2</sup> S clock generator architecture . . . . .	783
Figure 305. Block diagram of STM32C0x1 MCU and Cortex®-M0+-level debug support . . . . .	804

# 1 Documentation conventions

## 1.1 General information

The STM32C0x1 devices have an Arm®<sup>(a)</sup> Cortex®-M0+ core.



## 1.2 List of abbreviations for registers

The following abbreviations<sup>(b)</sup> are used in register descriptions:

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

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

## 1.3 Glossary

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

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

## 1.4 Availability of peripherals

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

## 2 Memory and bus architecture

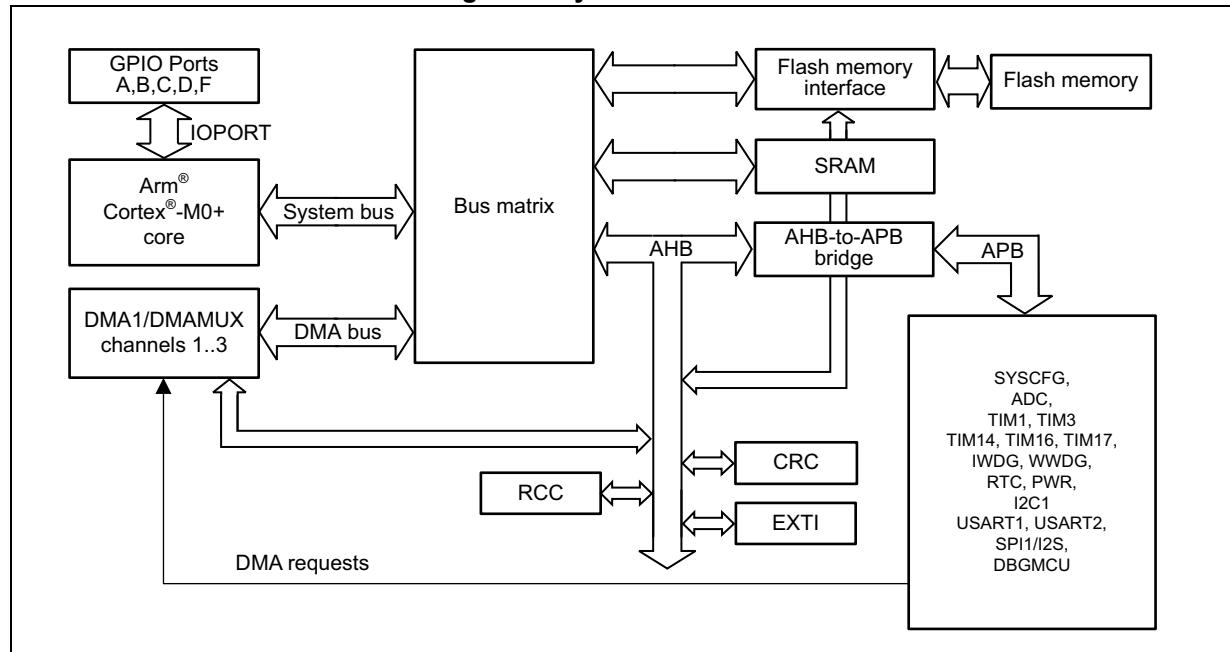
### 2.1 System architecture

The main system consists of:

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

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

**Figure 1. System architecture**



#### System bus (S-bus)

This bus connects the system bus of the Cortex®-M0+ core (peripheral bus) to a bus matrix that manages the arbitration between the core and the DMA.

#### DMA bus

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

## Bus matrix

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

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

## AHB-to-APB bridge (APB)

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

Refer to [Section 2.2: Memory organization](#) for the address mapping of the peripherals connected to this bridge.

After each device reset, all peripheral clocks are disabled (except for the SRAM and Flash memory). Before using a peripheral, its clock must first be enabled through the RCC\_AHBENR, RCC\_APBENRx, or RCC\_IOPENR 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

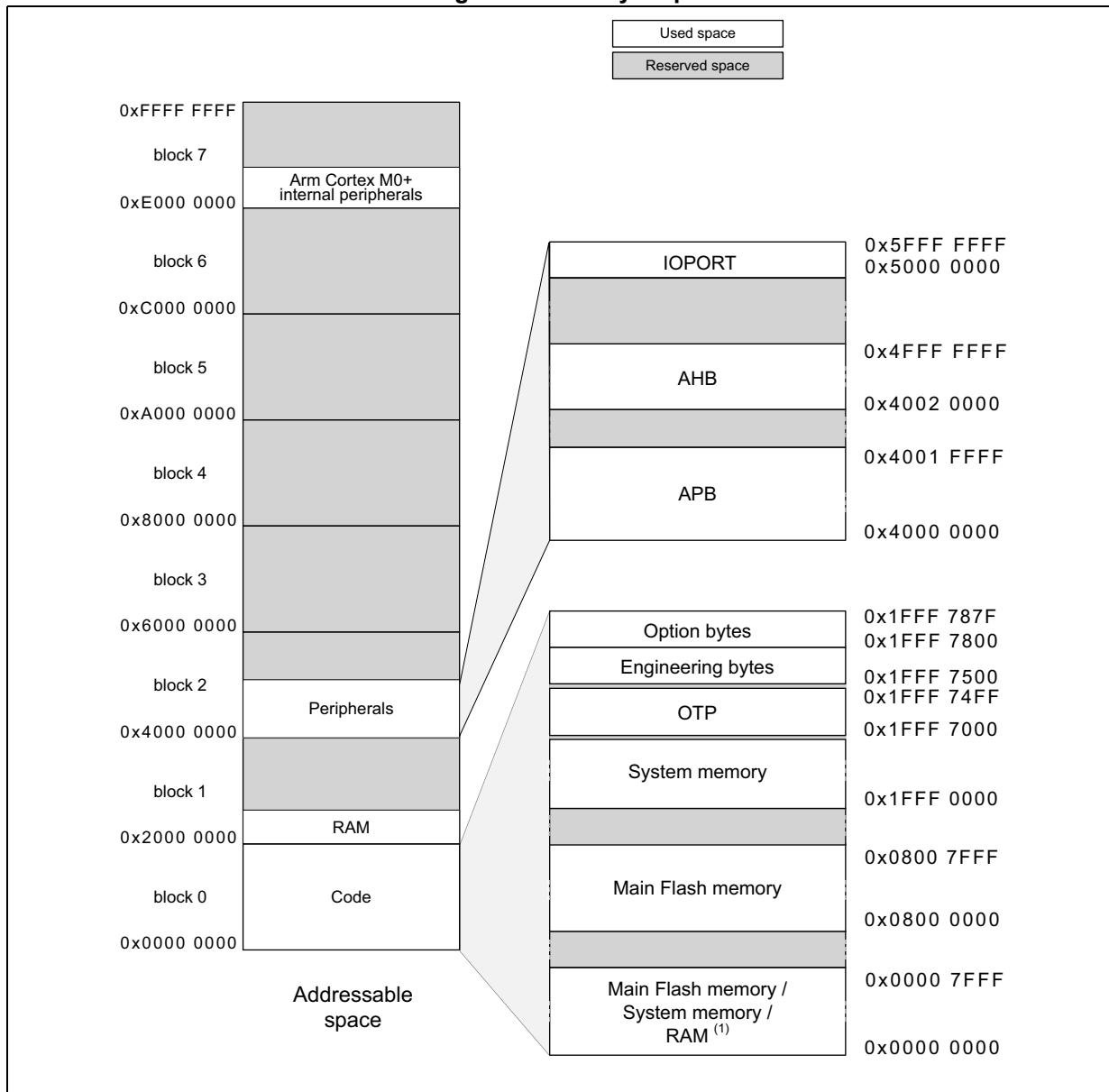
### 2.2.1 Introduction

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.

## 2.2.2 Memory map and register boundary addresses

Figure 2. Memory map



- Depends on boot configuration

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

Table 1. STM32C0x1 boundary addresses

Type	Boundary address	Size	Memory Area	Register description
SRAM	0x2000 3000 - 0x3FFF FFFF	~512 MB	Reserved	-
	0x2000 0000 - 0x2000 2FFF	12 KB	SRAM	-

**Table 1. STM32C0x1 boundary addresses (continued)**

Type	Boundary address	Size	Memory Area	Register description
Code	0xFFFF 7880 - 0xFFFF FFFF	~34 KB	Reserved	-
	0xFFFF 7800 - 0xFFFF 787F	128 B	Option bytes	<a href="#">Section 3.4 on page 55</a>
	0xFFFF 7500 - 0xFFFF 77FF	768 B	Engineering bytes	-
	0xFFFF 7400 - 0xFFFF 74FF	256 B	Reserved	-
	0xFFFF 7000 - 0xFFFF 73FF	1 KB	OTP	-
	0xFFFF 1800 - 0xFFFF 6FFF	~22 KB	Reserved	-
	0xFFFF 0000 - 0xFFFF 17FF	6 KB	System memory	-
	0x0800 8000 - 0xFFFF D7FF	~383 MB	Reserved	-
	0x0800 0000 - 0x0800 7FFF	32 KB	Main Flash memory	<a href="#">Section 3.3.1 on page 47</a>
	0x0000 8000 - 0x07FF FFFF	~127 MB	Reserved	-
	0x0000 0000 - 0x000 7FFF	32 KB	Main Flash memory, system memory or SRAM depending on BOOT configuration	-

**Table 2. STM32C0x1 peripheral register boundary addresses**

Bus	Boundary address	Size	Peripheral	Peripheral register map
-	0xE000 0000 - 0xE00F FFFF	1MB	Cortex®-M0+ internal peripherals	-
IOPORT	0x5000 1800 - 0x5FFF 17FF	~256 MB	Reserved	-
	0x5000 1400 - 0x5000 17FF	1 KB	GPIOF	<a href="#">Section 6.4.12 on page 160</a>
	0x5000 1000 - 0x5000 13FF	1 KB	Reserved	-
	0x5000 0C00 - 0x5000 0FFF	1 KB	GPIOD	<a href="#">Section 6.4.12 on page 160</a>
	0x5000 0800 - 0x5000 0BFF	1 KB	GPIOC	<a href="#">Section 6.4.12 on page 160</a>
	0x5000 0400 - 0x5000 07FF	1 KB	GPIOB	<a href="#">Section 6.4.12 on page 160</a>
	0x5000 0000 - 0x5000 03FF	1 KB	GPIOA	<a href="#">Section 6.4.12 on page 160</a>

**Table 2. STM32C0x1 peripheral register boundary addresses (continued)**

Bus	Boundary address	Size	Peripheral	Peripheral register map
AHB	0x4002 3400 - 0x4FFF FFFF	~256 MB	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 KB	CRC	<a href="#">Section 13.4.6 on page 237</a>
	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 KB	FLASH	<a href="#">Section 3.7.14 on page 79</a>
	0x4002 1C00 - 0x4002 1FFF	3 KB	Reserved	-
	0x4002 1800 - 0x4002 1BFF	1 KB	EXTI	<a href="#">Section 12.5.9 on page 229</a>
	0x4002 1400 - 0x4002 17FF	1 KB	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 KB	RCC	<a href="#">Section 5.4.22 on page 142</a>
	0x4002 0C00 - 0x4002 0FFF	1 KB	Reserved	-
	0x4002 0800 - 0x4002 0BFF	1 KB	DMAMUX	<a href="#">Section 10.6.7 on page 213</a>
	0x4002 0400 - 0x4002 07FF	1 KB	Reserved	-
	0x4002 0000 - 0x4002 03FF	1 KB	DMA1	<a href="#">Section 9.6.7 on page 197</a>
APB	0x4001 5C00 - 0x4001 FFFF	32 KB	Reserved	-
	0x4001 5800 - 0x4001 5BFF	1 KB	DBG	<a href="#">Section 26.10.5 on page 815</a>
	0x4001 4C00 - 0x4001 57FF	3 KB	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	<a href="#">Section 18.4.21 on page 543</a>
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	<a href="#">Section 18.4.21 on page 543</a>
	0x4001 4000 - 0x4001 43FF	1 KB	Reserved	-
	0x4001 3C00 - 0x4001 3FFF	1 KB	Reserved	-
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	<a href="#">Section 24.8.15 on page 744</a>
	0x4001 3400 - 0x4001 37FF	1 KB	Reserved	-
APB	0x4001 3000 - 0x4001 33FF	1 KB	SPI1/I2S1	<a href="#">Section 25.9.10 on page 803</a>
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	<a href="#">Section 15.4.29 on page 393</a>
	0x4001 2800 - 0x4001 2BFF	1 KB	Reserved	-
	0x4001 2400 - 0x4001 27FF	1 KB	ADC	<a href="#">Section 14.13 on page 295</a>
	0x4001 0400 - 0x4001 23FF	8 KB	Reserved	-
	0x4001 0080 - 0x4001 03FF	1 KB	SYSCFG(ITLINE) <sup>(1)</sup>	<a href="#">Section 7.1.25 on page 174</a>
	0x4001 001D - 0x4001 007F		Reserved	-
	0x4001 0000 - 0x4001 001C		SYSCFG	<a href="#">Section 7.1.25 on page 174</a>
	0x4000 B400 - 0x4000 FFFF	19 KB	Reserved	-
	0x4000 B000 - 0x4000 B3FF	1 KB	Reserved	-
	0x4000 8C00 - 0x4000 AFFF	9 KB	Reserved	-
	0x4000 8800 - 0x4000 8BFF	1 KB	Reserved	-
	0x4000 7400 - 0x4000 87FF	5 KB	Reserved	-
	0x4000 7000 - 0x4000 73FF	1 KB	PWR	<a href="#">Section 4.4.18 on page 101</a>

**Table 2. STM32C0x1 peripheral register boundary addresses (continued)**

Bus	Boundary address	Size	Peripheral	Peripheral register map
APB	0x4000 6000 - 0x4000 6FFF	4 KB	Reserved	-
	0x4000 5C00 - 0x4000 5FFF	1 KB	Reserved	-
	0x4000 5800 - 0x4000 5BFF	1 KB	Reserved	-
	0x4000 5400 - 0x4000 57FF	1 KB	I2C1	<a href="#">Section 23.7.12 on page 656</a>
	0x4000 5000 - 0x4000 53FF	1 KB	Reserved	-
	0x4000 4C00 - 0x4000 4FFF	1 KB	Reserved	-
	0x4000 4800 - 0x4000 4BFF	1 KB	Reserved	-
	0x4000 4400 - 0x4000 47FF	1 KB	USART2	<a href="#">Section 24.8.15 on page 744</a>
	0x4000 4000 - 0x4000 43FF	1 KB	Reserved	-
	0x4000 3C00 - 0x4000 3FFF	1 KB	Reserved	-
	0x4000 3800 - 0x4000 3BFF	1 KB	Reserved	-
	0x4000 3400 - 0x4000 37FF	1 KB	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 KB	IWDG	<a href="#">Section 20.4.6 on page 554</a>
	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG	<a href="#">Section 21.5.4 on page 560</a>
	0x4000 2800 - 0x4000 2BFF	1 KB	RTC	<a href="#">Section 22.6.18 on page 590</a>
	0x4000 2400 - 0x4000 27FF	1 KB	Reserved	-
	0x4000 2000 - 0x4000 23FF	1 KB	TIM14	<a href="#">Section 17.4.13 on page 491</a>
	0x4000 1800 - 0x4000 1FFF	2 KB	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 KB	Reserved	-
	0x4000 1000 - 0x4000 13FF	1 KB	Reserved	-
	0x4000 0C00 - 0x4000 0FFF	1 KB	Reserved	-
	0x4000 0800 - 0x4000 0BFF	1 KB	Reserved	-
	0x4000 0400 - 0x4000 07FF	1 KB	TIM3	<a href="#">Section 16.4.24 on page 465</a>
	0x4000 0000 - 0x4000 03FF	1 KB	Reserved	-

1. SYSCFG (ITLINE) registers use 0x4001 0000 as reference peripheral base address.

## 2.3 Embedded SRAM

The following table summarizes the SRAM resources on the devices, with parity check enabled and disabled.

**Table 3. SRAM size**

Device	SRAM with parity (Kbyte)
STM32C011xx	6
STM32C031xx	12

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

#### Parity check

The user can enable the parity check using the option bit RAM\_PARITY\_CHECK in the user option byte (refer to [Section 3.4: FLASH option bytes](#)).

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

The parity bits are computed and stored when writing into the SRAM. Then, they are automatically checked when reading. If one bit fails, an NMI is generated.

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

## 2.4 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block. It contains the application program and user data if necessary.
  - The information block. It is composed of three parts:
    - Option bytes for hardware and memory protection user configuration.
    - System memory which contains the proprietary boot loader code.
    - OTP (one-time programmable) area
- Refer to [Section 3: Embedded flash memory \(FLASH\)](#) for more details.

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

## 2.5 Boot configuration

In the STM32C0x1, three different boot modes can be selected through the BOOT0 pin and boot configuration bits nBOOT1, nBOOT\_SEL and nBOOT0 in the User option byte, as shown in the following table.

Table 4. Boot modes

Boot mode configuration					Selected boot area
BOOT_LOCK	nBOOT1 bit	BOOT0 pin	nBOOT_SEL bit	nBOOT0 bit	
0	x	0	0	x	Main Flash memory
0	1	1	0	x	System memory
0	0	1	0	x	Embedded SRAM
0	x	x	1	1	Main Flash memory
0	1	x	1	0	System memory

**Table 4. Boot modes (continued)**

Boot mode configuration					Selected boot area
BOOT_LOCK	nBOOT1 bit	BOOT0 pin	nBOOT_SEL bit	nBOOT0 bit	
0	0	x	1	0	Embedded SRAM
1	x	x	x	x	Main Flash memory

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

The boot mode configuration is also re-sampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the address stored in 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 0x1FFF0000.
- 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).

The BOOT\_LOCK bit allows forcing a unique entry point in the Main Flash memory for boot, regardless of the other boot mode configuration bits. See [Section 3.5.6: Forcing boot from flash memory](#).

### Empty check

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

This flag is updated only during Option bytes loading: it is set when the content of the address 0x0800 0000 is read as 0xFFFF FFFF, otherwise it is cleared. It means a power reset or setting of OBL\_LAUNCH bit in FLASH\_CR register is needed to clear this flag after programming of a virgin device to execute user code after System reset. The EMPTY bit can also directly be written by software.

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

## Physical remap

Once the boot mode is selected, the application software can modify the memory accessible in the code area. This modification is performed by programming the MEM\_MODE bits in the [\*SYSCFG configuration register 1 \(SYSCFG\\_CFGR1\)\*](#).

## Embedded boot loader

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

- USART1
- I2C1

For further details, refer to the device data sheets and the application note AN2606.

## 3 Embedded flash memory (FLASH)

### 3.1 FLASH Introduction

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

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

### 3.2 FLASH main features

- 32 Kbytes of flash memory (Main memory)
- Memory organization:
  - Main memory block: 32 Kbytes (4 Kbytes x 64 bits)
  - Information block: 10 Kbytes (1.28 Kbytes x 64 bits)
  - Page size: 2 Kbytes
- 64-bit wide data read (no ECC)
- Page erase (2 Kbytes) and mass erase

Flash memory interface features:

- Flash memory read operations
- Flash memory program/erase operations
- Read protection activated by option (RDP)
- Two write protection areas, selected by option (WRP)
- Two proprietary code read protection areas, selected by option (PCROP)
- Securable memory area
- Flash memory empty check
- Prefetch buffer
- CPU instruction cache: two cache lines of 64 bits (16-byte RAM)
- Option byte loader

### 3.3 FLASH functional description

#### 3.3.1 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 flash memory is organized as follows:

- Main memory block containing 16 pages of 2 Kbytes, each page with 8 rows of 256 bytes
- Information block containing:
  - System memory from which the CPU boots in System memory boot mode. The area is reserved and contains the boot loader used to reprogram the flash memory

through one of the following interfaces: USART1 and I2C1. On the manufacturing line, the devices are programmed and protected against spurious write/erase operations. For further details, refer to the AN2606 available from [www.st.com](http://www.st.com).

- 1 Kbyte (128 double words) OTP (one-time programmable) for user data. The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire double word (64 bits) cannot be written anymore, even with the value 0x0000 0000 0000 0000.
- The OTP area cannot be read when RDP level is 1 and boot source is not the Main flash memory area.
- Option bytes for user configuration.

The following table shows the mapping of the flash memory into Information block and Main memory area.

**Table 5. Flash memory organization**

Area	Addresses	Size (bytes)	Memory type
Information block	0x1FFF 7800 - 0x1FFF 7FFF	2K (only first 128 bytes used)	Option bytes
	0x1FFF 7400 - 0x1FFF 77FF	1 K	Engineering bytes
	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	0x1FFF 0000 - 0x1FFF 17FF	6 K	System memory
Main memory	0x0800 7800 - 0x0800 7FFF	2 K	Page 15
	...	...	...
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 0000 - 0x0800 07FF	2 K	Page 0

### 3.3.2 FLASH empty check

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

The Main flash memory empty check status can be read from the EMPTY bit in [\*FLASH access control register \(FLASH\\_ACR\)\*](#). Software can modify the Main flash memory empty status by writing an appropriate value to the EMPTY bit.

### 3.3.3 FLASH read access latency

To correctly read data from the flash memory, set the LATENCY[2:0] bitfield of the [\*FLASH access control register \(FLASH\\_ACR\)\*](#) register as per the following table.

**Table 6. LATENCY[2:0] setting as function of HCLK frequency**

HCLK (MHz)	LATENCY[2:0]
≤ 24	000 (1 HCLK cycle)
≤ 48	001 (2 HCLK cycles)

Upon power-on reset or upon wakeup from Standby, the HCLK clock frequency is automatically set to 12 MHz and the LATENCY[2:0] bitfield to 000. See [Section 5.2: Clocks](#).

To change HCLK frequency, respect the following sequence:

### Increasing HCLK frequency

1. Set the LATENCY[2:0] bitfield to correspond to the target HCLK frequency, as per [Table 6](#).
2. Read the LATENCY[2:0] bitfield until it returns the value written in the previous step.
3. Select the system clock source as required, through the SW[2:0] bitfield of the RCC\_CFG register.
4. Set the HCLK clock prescaler as required, through the HPRE[3:0] bitfield of the RCC\_CFG register.

The clock source effectively selected for the system can be checked by reading the clock source status bitfield SWS[2:0] of the RCC\_CFG register. The HPRE[3:0] bitfield of the RCC\_CFG register can also be read to check its content.

### Decreasing system clock frequency

1. Select the system clock source as required, through the SW[2:0] bitfield of the RCC\_CFG register.
2. Set the HCLK clock prescaler as required, through the HPRE[3:0] bitfield of the RCC\_CFG register.
3. Read the SWS[2:0] bitfield of the RCC\_CFG register until it returns the value set into SW[2:0] in step 1. The HPRE[3:0] bitfield of the RCC\_CFG register can also be read to check its content.
4. Set the LATENCY[2:0] bitfield to correspond to the target HCLK frequency, as per [Table 6](#).

## 3.3.4 Flash memory acceleration

### Instruction prefetch

Each flash memory read operation provides 64 bits from either two instructions of 32 bits or four instructions of 16 bits according to the program launched. This 64-bits current instruction line is saved in a current buffer. So, in case of sequential code, at least two CPU cycles are needed to execute the previous read instruction line. Prefetch on the CPU S-bus can be used to read the next sequential instruction line from the flash memory while the current instruction line is being requested by the CPU.

Prefetch is enabled by setting the PRFTEN bit of the [FLASH access control register \(FLASH\\_ACR\)](#). This feature is useful if at least one wait state is needed to access the flash memory.

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

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

### Cache memory

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

The Instruction cache memory is enabled after system reset.

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

## 3.3.5 FLASH program and erase operations

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

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

In contrast to the ICP method, **in-application programming (IAP)** can use any communication interface supported by the microcontroller (I/Os, UART, 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 success of a data word programming operation and a page/bank erase operation is not guaranteed if aborted due to device reset or power loss.

During a program/erase operation to the flash memory, any attempt to read the flash memory stalls the bus. The read operation proceeds correctly once the program/erase operation has completed.

### Unlocking the flash memory

After reset, write into the *FLASH control register (FLASH\_CR)* is not allowed so as to protect the flash memory against possible unwanted operations due, for example, to electric disturbances. The following sequence unlocks these registers:

1. Write KEY1 = 0x4567 0123 in the *FLASH key register (FLASH\_KEYR)*
2. Write KEY2 = 0xCDEF 89AB in the *FLASH key register (FLASH\_KEYR)*.

Any wrong sequence locks the FLASH\_CR registers until the next system reset. In the case of a wrong key sequence, a bus error is detected and a Hard Fault interrupt is generated.

The FLASH\_CR registers can be locked again by software by setting the LOCK bit in one of these registers.

**Note:** *The FLASH\_CR register cannot be written when the BSY1 bit of the [FLASH status register \(FLASH\\_SR\)](#) is set. Any attempt to write to this register with the BSY1 bit set causes the AHB bus to stall until the BSY1 bit is cleared.*

### 3.3.6 FLASH Main memory erase sequences

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

#### Flash memory page erase

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

**Table 7. Page erase overview**

SEC_PROT	PCROP	WRP	PCROP_RDP	Comment	WRPERR	CPU bus error
0	No	No	X	Page is erased	No	No
	No	Yes		Page erase aborted (no page erase started)	Yes	
	Yes	No			No	
	Yes	Yes				
1	X			Protected pages only	No	Yes

To erase a page (2 Kbytes), follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY1 bit of the [FLASH status register \(FLASH\\_SR\)](#).
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PER bit and select the page to erase (PNB) in the [FLASH control register \(FLASH\\_CR\)](#).
4. Set the STRT bit of the [FLASH control register \(FLASH\\_CR\)](#).
5. Wait until the BSY1 bit of the [FLASH status register \(FLASH\\_SR\)](#) is cleared.

**Note:** *The HSI48 internal oscillator (with a divide by three to provide 16 MHz) is automatically enabled when the STRT bit is set. It is automatically disabled when the STRT bit is cleared, except if previously enabled with the HSION bit of the RCC\_CR register.*

#### Flash memory bank or mass erase

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

**Table 8. Mass erase overview**

<b>SEC_PROT</b>	<b>PCROP</b>	<b>WRP</b>	<b>PCROP_RDP</b>	<b>Comment</b>	<b>WRPERR</b>	<b>CPU bus error</b>
0	No	No	X	Memory is erased	No	No
	No	Yes		Erase aborted (no erase started)	Yes	
	Yes	No			No	
	Yes	Yes				
1	X			Erase aborted (no erase started)	No	Yes

To perform a mass erase, respect the following procedure:

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

**Note:** *The HSI48 internal oscillator (with a divide by three to provide 16 MHz) is automatically enabled when the STRT bit is set. It is automatically disabled when the STRT bit is cleared, except if previously enabled with the HSION bit of the RCC\_CR register.*

### 3.3.7 FLASH Main memory programming sequences

The flash memory is programmed 64 bits at a time.

Programming a previously programmed address with a non-zero data is not allowed. Any such attempt sets PROGERR flag of the *FLASH status register (FLASH\_SR)*.

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

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

#### Standard programming

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

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

**Note:**

*When the flash memory interface has received a good sequence (a double word), programming is automatically launched and the BSY1 bit set.*

*The HSI48 internal oscillator (with a divide by three to provide 16 MHz) is automatically enabled when the PG bit is set. It is automatically disabled when the PG bit is cleared, except if previously enabled with the HSION bit of the RCC\_CR register.*

### Fast programming

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

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

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

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

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

1. Perform a mass or page erase. If not, PGSERR is set.
2. Check that no Main flash memory operation is ongoing by checking the BSY1 bit of the *FLASH status register (FLASH\_SR)*..
3. Check and clear all error programming flag due to a previous programming.
4. Set the FSTPG bit in *FLASH control register (FLASH\_CR)*.
5. Write 16 double-words to program a row (128 bytes).
6. Wait until the BSY1 bit of the *FLASH status register (FLASH\_SR)* is cleared.
7. Check that the EOP flag in the *FLASH status register (FLASH\_SR)* is set (programming operation succeeded), and clear it by software.
8. Clear the FSTPG bit of the *FLASH status register (FLASH\_SR)* if there are no more programming requests anymore.

**Note:**

*When attempting to write in Fast programming mode while a read operation is ongoing, the programming is aborted without any system notification (no error flag is set).*

*When the flash memory interface has received the first double word, programming is automatically launched. The BSY1 bit is set when the high voltage is applied for the first*

*double word, and it is cleared when the last double word has been programmed or in case of error.*

*The HSI48 internal oscillator (with a divide by three to provide 16 MHz) is automatically enabled when the FSTPG bit is set. It is automatically disabled when the FSTPG bit is cleared, except if previously enabled with the HSION bit of the RCC\_CR register.*

*The 16 double words must be written successively. The high voltage is kept on the flash memory for all the programming. Maximum time between two double words write requests is the time programming (around 20 µs). If a second double word arrives after this time programming, fast programming is interrupted and MISSERR is set.*

*High voltage must not exceed 8 ms for a full row between two erases. This is guaranteed by the sequence of 16 double words successively written with a clock system greater or equal to 8 MHz. An internal time-out counter counts 7 ms when Fast programming is set and stops the programming when time-out is over. In this case the FASTERR bit is set.*

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

## Programming errors

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

- **PROGERR:** Programming error

In standard programming: PROGERR is set if the word to write is not previously erased (except if the value to program is full zero and the target address is in the Main memory).

- **SIZERR:** Size programming error

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

- **PGAERR:** Alignment programming error

PGAERR is set if one of the following conditions occurs:

- In standard programming: the first word to be programmed is not aligned with a double word address, or the second word doesn't belong to the same double word address.
- In fast programming: the data to program doesn't belong to the same row than the previous programmed double words, or the address to program is not greater than the previous one.

- **PGSERR:** Programming sequence error

PGSERR is set if one of the following conditions occurs:

- In the standard programming sequence or the fast programming sequence: a data is written when PG and FSTPG are cleared.
- In the standard programming sequence or the fast programming sequence: MER1 and PER are not cleared when PG or FSTPG is set.
- In the fast programming sequence: the Mass erase is not performed before setting the FSTPG bit.
- In the mass erase sequence: PG, FSTPG, and PER are not cleared when MER1 is set.
- In the page erase sequence: PG, FSTPG and MER1 are not cleared when PER is set.

- PGSERR is set also if PROGERR, SIZERR, PGAERR, WRPERR, MISSERR, FASTERR or PGSERR is set due to a previous programming error.
- **WRPERR:** Write protection error  
WRPERR is set if one of the following conditions occurs:
  - Attempt to program or erase in a write protected area (WRP) or in a PCROP area.
  - Attempt to perform a mass erase when one page or more is protected by WRP or PCROP.
  - The debug features are connected or the boot is executed from SRAM or from system flash memory when the read protection (RDP) is set to Level 1.
  - Attempt to modify the option bytes when the read protection (RDP) is set to Level 2.
- **MISSERR:** Fast programming data miss error  
In fast programming: all the data must be written successively. MISSERR is set if the previous data programmation is finished and the next data to program is not written yet.
- **FASTERR:** Fast programming error  
In fast programming: FASTERR is set if one of the following conditions occurs:
  - when FSTPG bit is set for more than 8 ms, which generates a time-out detection
  - when the row fast programming has been interrupted by a MISSERR, PGAERR, WRPERR or SIZERR

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

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

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

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

### Programming and cache

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

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

## 3.4 FLASH option bytes

### 3.4.1 FLASH option byte description

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

A double word is split up in option bytes as indicated in [Table 9](#).

**Table 9. Option byte format**

<b>63-56</b>	<b>55-48</b>	<b>47-40</b>	<b>39-32</b>	<b>31-24</b>	<b>23-16</b>	<b>15 -8</b>	<b>7-0</b>
Complemented option byte 3	Complemented option byte 2	Complemented option byte 1	Complemented option byte 0	Option byte 3	Option byte 2	Option byte 1	Option byte 0

[Table 10](#) shows the organization of the option bytes (the lower word only) in the flash memory information block. The software can read the option bytes from these flash memory locations or from their corresponding option registers referenced in the table. Refer to sections [3.7.6](#) to [3.7.13](#) for the description of the option register bitfields, also applicable to the option byte bitfields.

**Table 10. Organization of option bytes**

1. The upper 32 bits of the double-word address contain the inverted data from the lower 32 bits.
  2. Only relevant to 48-pin packages that are only available for STM32C031xx.

### 3.4.2 FLASH option byte programming

After reset, the option related bits of the [\*FLASH control register \(FLASH\\_CR\)\*](#) are write-protected. To run any operation on the option byte page, the option lock bit OPTLOCK of the [\*FLASH control register \(FLASH\\_CR\)\*](#) must be cleared. The following sequence is used to unlock this register:

1. Unlock the FLASH\_CR with the LOCK clearing sequence (refer to [\*Unlocking the flash memory\*](#))
2. Write OPTKEY1 = 0x0819 2A3B of the [\*FLASH option key register \(FLASH\\_OPTKEYR\)\*](#)
3. Write OPTKEY2 = 0x4C5D 6E7F of the [\*FLASH option key register \(FLASH\\_OPTKEYR\)\*](#)

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

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

**Note:**

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

#### Modifying user options

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

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

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

**Note:**

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

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

**Caution:**

Upon an option byte programming failure (for any reason, such as loss of power or a reset during the option byte change sequence), the mismatch values of the option bytes are loaded after reset. Those mismatch values force a secure configuration that might permanently lock the device. To prevent this, only program option bytes in a safe environment – safe supply, no pending watchdog, and clean reset line.

#### Option byte loading

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

Option byte loading is performed in two cases:

- when OBL\_LAUNCH bit of the *FLASH control register (FLASH\_CR)* is set
- after a power reset (BOR reset or exit from Standby/Shutdown modes)

Option byte loader performs a read of the options block and stores the data into internal option registers. These internal registers configure the system and can be read by software. Setting OBL\_LAUNCH generates a reset so the option byte loading is performed under system reset.

Each option bit has also its complement in the same double word. During option loading, a verification of the option bit and its complement allows to check the loading has correctly taken place.

During option byte loading, the options are read by double word.

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

If the comparison between the word and its complement fails, a status bit OPTVERR is set. Mismatch values are forced into the option registers:

- For USR OPT option, the value of mismatch is 1 for all option bits, except the BOR\_EN bit that is 0 (BOR disabled).
- For WRP option, the value of mismatch is the default value “No protection”.
- For RDP option, the value of mismatch is the default value “Level 1”.
- For PCROP, the value of mismatch is “all memory protected”.
- For BOOT\_LOCK, the value of mismatch is “boot forced from Main flash memory”.

Upon system reset, the option bytes are copied into the following option registers that can be read and written by software:

- FLASH\_OPTR
- FLASH\_PCROP1xSR ( $x = A$  or  $B$ )
- FLASH\_PCROP1xER ( $x = A$  or  $B$ )
- FLASH\_WRP1xR ( $x = A$  or  $B$ )
- FLASH\_SECR

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

## 3.5 Flash memory protection

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

### 3.5.1 FLASH read protection (RDP)

The read protection is activated by setting the RDP option byte and then, by applying a system reset to reload the new RDP option byte. The read protection protects the Main flash memory, the option bytes, and the SRAM.

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

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

The flash memory is protected when the RDP option byte and its complement contain the pair of values shown in [Table 11](#).

**Table 11. Flash memory read protection status**

RDP byte value	RDP complement byte value	Read protection level
0xAA	0x55	Level 0
Any values except the combinations [0xAA, 0x55] and [0xCC, 0x33]		Level 1 (default)
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 within the Main flash memory area are possible. The option bytes are also accessible by all operations.

### Level 1: Read protection

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

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

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

### Level 2: No debug

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

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

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

## Changing the read protection level

The read protection level can change:

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

Once in Level 2, it is no more possible to modify the read protection level.

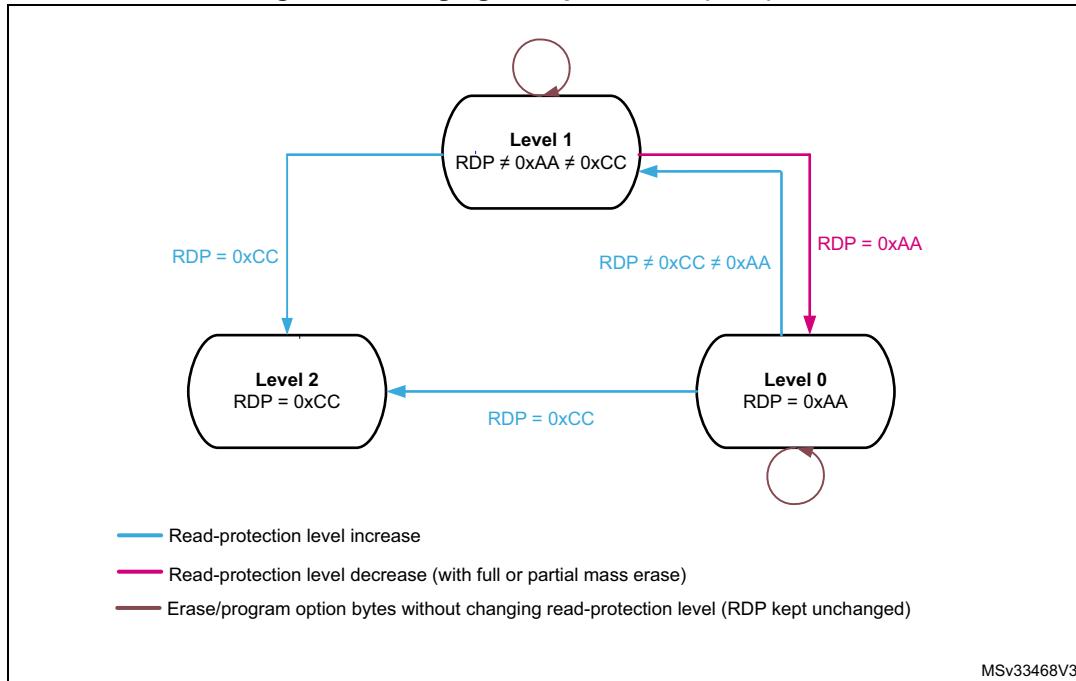
With the PCROP\_RDP bit of the [FLASH PCROP area A end address register \(FLASH\\_PCROP1AER\)](#) set, the change from Level 1 to Level 0 triggers full mass erase of the Main flash memory. The user options except PCROP protection are set to their previous values copied from FLASH\_OPTR and FLASH\_WRP1xR (x = A or B). PCROP is disabled. The OTP area is not affected by mass erase and remains unchanged.

With the PCROP\_RDP bit cleared, a partial mass erase occurs, only erasing flash memory pages that do not overlap with PCROP area (do not contain any PCROP-protected subpage). The option bytes are re-programmed with their previous values. This is also true for FLASH\_PCROP1xSR and FLASH\_PCROP1xER registers (x = A or B).

**Table 12: Mass erase upon RDP regression from Level 1 to Level 0**

PCROP area	PCROP_RDP	Mass erase
None	x	Full
Part of flash memory	1	Partial (flash memory pages not overlapping with PCROP area)
	0	None
Full flash memory		

**Note:** *Mass erase (full or partial) is only triggered by the RDP regression from Level 1 to Level 0. RDP level increase (Level 0 to Level 1, 1 to 2, or 0 to 2) does not cause any mass erase.*  
*To validate the protection level change, the option bytes must be reloaded by setting the OBL\_LAUNCH bit of the [FLASH control register \(FLASH\\_CR\)](#).*

**Figure 3. Changing read protection (RDP) level****Table 13. Access status versus protection level and execution modes**

Area	Protection level	User execution (BootFromFlash)			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	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
Option bytes	1	Yes	Yes <sup>(3)</sup>	Yes	Yes	Yes <sup>(3)</sup>	Yes
	2	Yes	No	No	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
OTP	1	Yes	Yes	N/A	No	No	N/A
	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 Flash Main memory is erased when the RDP option byte is programmed with all level of protections disabled (0xAA).

### 3.5.2 FLASH proprietary code readout protection (PCROP)

Two areas of the flash memory can be protected against unwanted read and/or write by a third party.

The protected area is execute-only: it can only be reached by the STM32 CPU, with an instruction code, while all other accesses (DMA, debug and CPU data read, write and erase) are strictly prohibited. The PCROP areas have subpage (512-byte) granularity. An additional option bit (PCROP\_RDP) allows to select if the PCROP area is erased or not

when the RDP protection is changed from Level 1 to Level 0 (refer to [Changing the read protection level](#)).

Each PCROP area is defined by a start subpage offset and an end subpage offset into the flash memory. These offsets are defined with the corresponding bitfields of the PCROP address registers *FLASH PCROP area A start address register (FLASH\_PCROP1ASR)*, *FLASH PCROP area A end address register (FLASH\_PCROP1AER)*, *FLASH PCROP area B start address register (FLASH\_PCROP1BSR)*, and *FLASH PCROP area B end address register (FLASH\_PCROP1BER)*.

A PCROP area *x* (A or B) is defined from the address:

$$\text{flash memory base address} + [\text{PCROP1x\_STRT} \times 0x200] \text{ (included)}$$

to the address:

$$\text{flash memory base address} + [(\text{PCROP1x\_END} + 1) \times 0x200] \text{ (excluded).}$$

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

$$\text{PCROP1x\_END} = \text{PCROP1x\_STRT} + 1.$$

When

$$\text{PCROP1x\_END} = \text{PCROP1x\_STRT},$$

the full flash memory is PCROP-protected.

For example, to PCROP-protect the address area from 0x0800 0800 to 0x0800 13FF, set the PCROP start subpage bitfield of the FLASH\_PCROP1xSR register and the PCROP end subpage bitfield of the FLASH\_PCROP1xER register (*x* = A or B) as follows:

- PCROP1x\_STRT = 0x04 (PCROP area start address 0x0800 0800)
- PCROP1x\_END = 0x09 (PCROP area end address 0x0800 13FF)

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

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

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

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

With the option bit PCROP\_RDP cleared, the change of RDP from Level 1 to Level 0 triggers a partial mass erase that preserves the contents of the flash memory pages overlapping with PCROP-protected areas. Refer to section [Changing the read protection level](#) for details.

**Table 14: PCROP protection**

PCROP register values ( <i>x</i> = A or B)	PCROP-protected area
PCROP1x_STRT = PCROP1x_END	Full flash memory

**Table 14: PCROP protection (continued)**

PCROP register values (x = A or B)	PCROP-protected area
PCROP1x_STRT > PCROP1x_END	None (unprotected)
PCROP1x_STRT < PCROP1x_END	Subpages from PCROP1x_STRT to PCROP1x_END (read-, write-, and erase-protected); PCROP area boundary pages (erase-protected).

**Note:** With PCROP\_RDP cleared, it is recommended to either define the PCROP area start and end onto flash memory page boundaries (2-Kbyte granularity), or to keep reserved and empty the PCROP-unprotected memory space of the PCROP area boundary pages (pages inside which the PCROP area starts and ends).

### 3.5.3 FLASH write protection (WRP)

The user area in flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined, with page (2-Kbyte) granularity. Each area is defined by a start page offset and an end page offset related to the physical flash memory base address. These offsets are defined in the WRP address registers [FLASH WRP area A address register \(FLASH\\_WRP1AR\)](#) and [FLASH WRP area B address register \(FLASH\\_WRP1BR\)](#).

The WRP *x* area (*x* = A, B) is defined from the address

$$\text{flash memory Base address} + [\text{WRP1x\_STRT} \times 0x0800] \text{ (included)}$$

to the address

$$\text{flash memory Base address} + [(\text{WRP1x\_END}+1) \times 0x0800] \text{ (excluded).}$$

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

$$\text{WRP1x\_END} = \text{WRP1x\_STRT}.$$

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

If boot in flash memory is selected, FLASH\_WRP1AR register must be programmed with:

- WRP1A\_STRT = 0x02.
- WRP1A\_END = 0x07.

WRP1B\_STRT and WRP1B\_END in FLASH\_WRP1BR can be used instead (area B in the flash memory).

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

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

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

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

**Table 15: WRP protection**

WRP registers values (x = A or B)	WRP-protected area
WRP1x_STRT = WRP1x_END	Page WRP1x
WRP1x_STRT > WRP1x_END	None (unprotected)
WRP1x_STRT < WRP1x_END	Pages from WRP1x_STRT to WRP1x_END

**Note:** To validate the WRP options, the option bytes must be reloaded by setting the OBL\_LAUNCH bit in flash memory control register.

### 3.5.4 Securable memory area

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

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

The size of the securable memory area is defined by the SEC\_SIZE[4:0] bitfield of the FLASH\_SECR register. It can be modified only in RDP Level 0. Its content is erased upon changing from RDP Level 1 to Level 0, even if it overlaps with PCROP subpages.

**Note:** The securable memory area start address is 0x0800 0000. Before activating the securable memory area, move the vector table outside the page 0 if necessary.

**Note:** Upon change from RDP Level 1 to Level 0 while the PCROP\_RDP bit is cleared, the securable memory area is erased even if it overlaps with the PCROP subpages. The PCROP subpages not overlapping with the securable memory area are not erased. See [Table 16](#).

**Table 16. Securable memory erase at RDP Level 1 to Level 0 change**

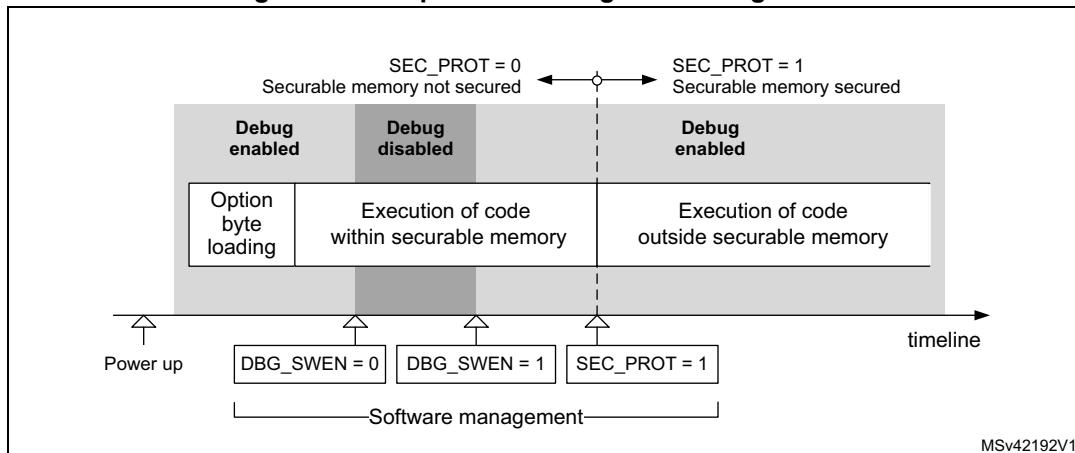
Securable memory size (SEC_SIZE[4:0])	PCROP_RDP	Erased pages
0	1	All (mass erase)
0	0	All but PCROP
> 0	1	All (mass erase)
> 0	0	All but PCROP outside the securable memory area

### 3.5.5 Disabling core debug access

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

*Figure 4* gives an example of managing DBG\_SWEN and SEC\_PROT bits.

**Figure 4. Example of disabling core debug access**



### 3.5.6 Forcing boot from flash memory

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

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

## 3.6 FLASH interrupts

**Table 17. FLASH interrupt requests**

Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit
End of operation	EOP <sup>(1)</sup>	Write EOP=1	EOPIE
Operation error	OPERR <sup>(2)</sup>	Write OPERR=1	ERRIE
Read protection error	RDERR	Write RDERR=1	RDERRIE
Write protection error	WRPERR	Write WRPERR=1	N/A
Size error	SIZERR	Write SIZERR=1	N/A
Programming sequential error	PROGERR	Write PROGERR=1	N/A
Programming alignment error	PGAERR	Write PGAERR=1	N/A
Programming sequence error	PGSERR	Write PGSERR=1	N/A

**Table 17. FLASH interrupt requests (continued)**

Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit
Data miss during fast programming error	MISSERR	Write MISSERR=1	N/A
Fast programming error	FASTERR	Write FASTERR=1	N/A

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

## 3.7 FLASH registers

### 3.7.1 FLASH access control register (FLASH\_ACR)

Address offset: 0x000

Reset value: 0x0004 0600

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.	DBG_SWEN	Res.	EMPTY
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.	Res.	Res.	Res.	ICRST	Res.	ICEN	PRFTEN	Res.	LATENCY[2:0]								
				rw		rw	rw								rw	rw	rw

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

Bit 18 **DBG\_SWEN**: Debug access software enable

Software may use this bit to enable/disable the debugger read access.

- 0: Debugger disabled
- 1: Debugger enabled

Bit 17 Reserved, must be kept at reset value.

Bit 16 **EMPTY**: Main flash memory area empty

This bit indicates whether the first location of the Main flash memory area is erased or has a programmed value.

- 0: Main flash memory area programmed
  - 1: Main flash memory area empty
- The bit can be set and reset by software.

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

Bit 11 **ICRST**: CPU Instruction cache reset

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

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ICEN**: CPU Instruction cache enable

- 0: CPU Instruction cache is disabled
- 1: CPU Instruction cache is enabled

Bit 8 **PRFTEN**: CPU Prefetch enable

- 0: CPU Prefetch disabled
- 1: CPU Prefetch enabled

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

Bits 2:0 **LATENCY[2:0]**: Flash memory access latency

The value in this bitfield represents the number of CPU wait states when accessing the flash memory.

- 000: Zero wait states
- 001: One wait state
- Other: Reserved

A new write into the bitfield becomes effective when it returns the same value upon read.

### 3.7.2 FLASH key register (FLASH\_KEYR)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: FLASH key

The following values must be written consecutively to unlock the *FLASH control register (FLASH\_CR)*, thus enabling programming/erasing operations:

KEY1: 0x4567 0123

KEY2: 0xCDEF 89AB

### 3.7.3 FLASH option key register (FLASH\_OPTKEYR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:0 **OPTKEY[31:0]**: Option byte key

The following values must be written consecutively to unlock the flash memory option registers, enabling option byte programming/erasing operations:

KEY1: 0x0819 2A3B

KEY2: 0x4C5D 6E7F

### 3.7.4 FLASH status register (FLASH\_SR)

Address offset: 0x010

Reset value: 0x000X 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.	CFGBSY	Res.	BSY1
													r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTV ERR	RD ERR	Res.	Res.	Res.	Res.	FAST ERR	MISS ERR	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
rc_w1	rc_w1					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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

Bit 18 **CFGBSY**: Programming or erase configuration busy.

This flag is set and cleared by hardware (set when the first word is sent and cleared after the second word is sent when the flash memory program or erase operation completes or ends with an error).

When set, the programming and erase settings of the PG, PNB[3:0], PER, and MER1 bitfields in *FLASH control register (FLASH\_CR)* are used (busy) and cannot be changed (a programming or erase operation is ongoing).

When cleared, the programming and erase settings of the PG, PNB[3:0], PER, and MER1 bitfields in *FLASH control register (FLASH\_CR)* can be modified.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BSY1**: Busy

This flag indicates that a flash memory operation requested by *FLASH control register (FLASH\_CR)* is in progress. This bit is set at the beginning of the flash memory operation, and cleared when the operation finishes or when an error occurs.

Bit 15 **OPTVERR**: Option and Engineering bits loading validity error

Set by hardware when the options and engineering bits read may not be the one configured by the user or production. If options and engineering bits haven't been properly loaded, OPTVERR is set again after each system reset. Option bytes that fail loading are forced to a safe value, see *Section 3.4.2: FLASH option byte programming*.

Cleared by writing 1.

Bit 14 **RDERR**: PCROP read error

Set by hardware when an address to be read belongs to a read protected area of the flash memory (PCROP protection). An interrupt is generated if RDERRIE is set in FLASH\_CR.

Cleared by writing 1.

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

Bit 9 **FASTERR**: Fast programming error

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

Cleared by writing 1.

Bit 8 **MISSERR**: Fast programming data miss error

In Fast programming mode, 16 double words (128 bytes) must be sent to flash memory successively, and the new data must be sent to the logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.

Cleared by writing 1.

Bit 7 **PGSERR**: Programming sequence error

Set by hardware when a write access to the flash memory is performed by the code while PG or FSTPG have not been set previously. Set also by hardware when PROGERR, SIZERR, PGAERR, WRPERR, MISSERR or FASTERR is set due to a previous programming error.

Cleared by writing 1.

Bit 6 **SIZERR**: Size error

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

Bit 5 **PGAERR**: Programming alignment error

Set by hardware when the data to program cannot be contained in the same double word (64-bit) flash memory in case of standard programming, or if there is a change of page during fast programming.

Cleared by writing 1.

Bit 4 **WRPERR**: Write protection error

Set by hardware when an address to be erased/programmed belongs to a write-protected part (by WRP, PCROP or RDP Level 1) of the flash memory.

Cleared by writing 1.

Bit 3 **PROGERR**: Programming error

Set by hardware when a double-word address to be programmed contains a value different from '0xFFFF FFFF' before programming, except if the data to write is '0x0000 0000'.

Cleared by writing 1.

## Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Operation error

Set by hardware when a flash memory operation (program / erase) completes unsuccessfully.

This bit is set only if error interrupts are enabled (ERRIE=1).

Cleared by writing '1'.

Bit 0 **EOP**: End of operation

Set by hardware when one or more flash memory operation (programming / erase) has been completed successfully.

This bit is set only if the end of operation interrupts are enabled (EOPIE=1).

Cleared by writing 1.

### 3.7.5 FLASH control register (FLASH\_CR)

Address offset: 0x014

Reset value: 0xC000 0000

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res.	SEC PROT	OBL LAUNCH	RD ERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPT STRT	STRT
rs	rs		rw	rc_w1	rw	rw	rw						rw	rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PNB[3:0]				MER1	PER	PG								
									rw	rw	rw	rw	rw	rw	rw

**Bit 31 **LOCK**: FLASH\_CR Lock**

This bit is set only. When set, the FLASH\_CR register is locked. It is cleared by hardware after detecting the unlock sequence.

In case of an unsuccessful unlock operation, this bit remains set until the next system reset.

**Bit 30 **OPTLOCK**: Options Lock**

This bit is set only. When set, all bits concerning user option in FLASH\_CR register and so option page are locked. This bit is cleared by hardware after detecting the unlock sequence. The LOCK bit must be cleared before doing the unlock sequence for OPTLOCK bit.

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

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

**Bit 28 **SEC\_PROT**: Securable memory area protection enable**

This bit enables the protection on securable area, provided that a non-null securable memory area size (SEC\_SIZE[4:0]) is defined in option bytes.

0: Disable (securable area accessible)

1: Enable (securable area not accessible)

This bit is possible to set only by software and to clear only through a system reset.

**Bit 27 **OBL\_LAUNCH**: Option byte load launch**

When set, this bit triggers the load of option bytes into option registers. It is automatically cleared upon the completion of the load. The high state of the bit indicates pending option byte load.

The bit cannot be cleared by software. It cannot be written as long as OPTLOCK is set.

**Bit 26 **RDERRIE**: PCROP read error interrupt enable**

This bit enables the interrupt generation upon setting the RDERR flag in the FLASH\_SR register.

0: Disable

1: Enable

**Bit 25 **ERRIE**: Error interrupt enable**

This bit enables the interrupt generation upon setting the OPERR flag in the FLASH\_SR register.

0: Disable

1: Enable

**Bit 24 **EOPIE**: End-of-operation interrupt enable**

This bit enables the interrupt generation upon setting the EOP flag in the FLASH\_SR register.

0: Disable

1: Enable

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

**Bit 18 **FSTPG**: Fast programming enable**

0: Disable

1: Enable

**Bit 17 **OPTSTRT**: Start of modification of option bytes**

This bit triggers an options operation when set.

This bit is set only by software, and is cleared when the BSY1 bit is cleared in FLASH\_SR.

Bit 16 **STRT**: Start erase operation

This bit triggers an erase operation when set.

This bit is possible to set only by software and to clear only by hardware. The hardware clears it when one of BSY1 and BSY2 flags in the FLASH\_SR register transits to zero.

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

Bits 6:3 **PNB[3:0]**: Page number selection

These bits select the page to erase:

0x0: page 0

0x1: page 1

...

0xF: page 15

*Note: Values corresponding to addresses outside the Main memory are not allowed.*

Bit 2 **MER1**: Mass erase

When set, this bit triggers the mass erase, that is, all user pages.

Bit 1 **PER**: Page erase enable

0: Disable

1: Enable

Bit 0 **PG**: Flash memory programming enable

0: Disable

1: Enable

### 3.7.6 FLASH option register (FLASH\_OPTR)

Address offset: 0x020

Reset value: 0xFFFFFFFF (The option bits are loaded with values from flash memory at power-on reset release.)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IRHEN	NRST_MODE [1:0]		n BOOT0	n BOOT1	nBOOT_SEL	SECURE_MUX_EN	RAM_P_ARITY_CHECK	HSE_NOT_REMAPPED	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nRST_SHDW	nRST_STDBY	nRST_STOP	BORFLEV[1:0]		BORRLEV[1:0]		BOR_EN	RDP[7:0]							
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.

Bit 29 **IRHEN**: Internal reset holder enable bit

0: Internal resets are propagated as simple pulse on NRST pin

1: Internal resets drives NRST pin low until it is seen as low level

- Bits 28:27 **NRST\_MODE[1:0]**: NRST pin configuration
- 00: Reserved
  - 01: Reset input only: a low level on the NRST pin generates system reset; internal RESET is not propagated to the NRST pin.
  - 10: Standard GPIO: only internal RESET is possible
  - 11: Bidirectional reset: the NRST pin is configured in reset input/output (legacy) mode
- Bit 26 **nBOOT0**: nBOOT0 option bit
- 0: nBOOT0 = 0
  - 1: nBOOT0 = 1
- Bit 25 **nBOOT1**: Boot configuration
- Together with the BOOT0 pin or option bit nBOOT0 (depending on nBOOT\_SEL option bit configuration), this bit selects boot mode from the Main flash memory, SRAM or the System memory. Refer to [Section 2.5: Boot configuration](#).
- Bit 24 **nBOOT\_SEL**: BOOT0 signal source selection
- This option bit defines the source of the BOOT0 signal.
- 0: BOOT0 pin (legacy mode)
  - 1: nBOOT0 option bit
- Bit 23 **SECURE\_MUXING\_EN**: Multiple-bonding security
- The bit allows enabling automatic I/O configuration to prevent conflicts on I/Os connected (bonded) onto the same pin.
- 0: Disable
  - 1: Enable
- If the software sets one of the I/Os connected to the same pin as active by configuring the SYSCFG\_CFGR3 register, enabling this bit automatically forces the other I/Os in digital input mode, regardless of their software configuration.
- When the bit is disabled, the SYSCFG\_CFGR3 register setting is ignored, all GPIOs linked to a given pin are active and can be set in the mode specified by the corresponding GPIOx\_MODER register. The user software must ensure that there is no conflict between GPIOs.
- Bit 22 **RAM\_PARITY\_CHECK**: SRAM parity check control enable/disable
- 0: Enable
  - 1: Disable
- Bit 21 **HSE\_NOT\_REMAPPED**: HSE remapping enable/disable
- When cleared, the bit remaps the HSE clock source from PF0-OSC\_IN/PF1-OSC\_OUT pins to PC14-OSCX\_IN/PC15-OSCX\_OUT. Thus PC14-OSCX\_IN/PC15-OSCX\_OUT are shared by both LSE and HSE and the two clock sources cannot be used simultaneously.
- 0: Enable
  - 1: Disable
- On packages with less than 48 pins, the remapping is always enabled (PF0-OSC\_IN/PF1-OSC\_OUT are not available), regardless of this bit. As all STM32C011xx packages have less than 48 pins, this bit is only applicable to STM32C031xx.
- Note: On 48 pins packages, when HSE\_NOT\_REMAPPED is reset, HSE cannot be used in bypass mode. Refer to product errata sheet for more details.*
- Bit 20 Reserved, must be kept at reset value.
- Bit 19 **WWDG\_SW**: Window watchdog selection
- 0: Hardware window watchdog
  - 1: Software window watchdog

- Bit 18 **IWDG\_STDBY**: Independent watchdog counter freeze in Standby mode  
0: Independent watchdog counter is frozen in Standby mode  
1: Independent watchdog counter is running in Standby mode
- Bit 17 **IWDG\_STOP**: Independent watchdog counter freeze in Stop mode  
0: Independent watchdog counter is frozen in Stop mode  
1: Independent watchdog counter is running in Stop mode
- Bit 16 **IDWG\_SW**: Independent watchdog selection  
0: Hardware independent watchdog  
1: Software independent watchdog
- Bit 15 **nRSTS\_SHDW**  
0: Reset generated when entering the Shutdown mode  
1: No reset generated when entering the Shutdown mode
- Bit 14 **nRST\_STDBY**  
0: Reset generated when entering the Standby mode  
1: No reset generated when entering the Standby mode
- Bit 13 **nRST\_STOP**  
0: Reset generated when entering the Stop mode  
1: No reset generated when entering the Stop mode
- Bits12:11 **BORF\_LEV[1:0]**: BOR threshold at falling V<sub>DD</sub> supply  
Falling V<sub>DD</sub> crossings this threshold activates the reset signal.  
00: BOR falling level 1 with threshold around 2.0 V  
01: BOR falling level 2 with threshold around 2.2 V  
10: BOR falling level 3 with threshold around 2.5 V  
11: BOR falling level 4 with threshold around 2.8 V
- Bits10:9 **BORR\_LEV[1:0]**: BOR threshold at rising V<sub>DD</sub> supply  
Rising V<sub>DD</sub> crossings this threshold releases the reset signal.  
00: BOR rising level 1 with threshold around 2.1 V  
01: BOR rising level 2 with threshold around 2.3 V  
10: BOR rising level 3 with threshold around 2.6 V  
11: BOR rising level 4 with threshold around 2.9 V
- Bit 8 **BOR\_EN**: Brown out reset enable  
0: Configurable brown out reset disabled, power-on reset defined by POR/PDR levels  
1: Configurable brown out reset enabled, values of BORR\_LEV and BORF\_LEV taken into account
- Bits 7:0 **RDP[7:0]**: Read protection level  
0xAA: Level 0, read protection not active  
0xCC: Level 2, chip read protection active  
Other: Level 1, memories read protection active

### 3.7.7 FLASH PCROP area A start address register (FLASH\_PCROP1ASR)

Address offset: 0x024

Reset value: 0b0000 0000 0000 0000 0000 00XX XXXX (The option bits are loaded with values from flash memory at power-on reset release.)

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

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
PCROP1A_STRT[5:0]															
												rw	rw	rw	rw

Bits 31:6 Reserved, must be kept cleared

Bits 5:0 **PCROP1A\_STRT[5:0]**: PCROP1A area start offset

Contains the offset of the first subpage of the PCROP1A area.

*Note: The number of effective bits depends on the size of the flash memory in the device.*

### 3.7.8 FLASH PCROP area A end address register (FLASH\_PCROP1AER)

Address offset: 0x028

Reset value: 0bX000 0000 0000 0000 0000 00XX XXXX (The option bits are loaded with values from flash memory at power-on reset release.)

Access: no wait state when no flash memory operation is on going, word, half-word access.  
PCROP\_RDP bit can be accessed with byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCROP_RDP	Res.														
rs															
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.	Res.	Res.
PCROP1A_END[5:0]															
												rw	rw	rw	rw

**Bit 31 PCROP\_RDP:** PCROP area erase upon RDP level regression

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

0: Not erased

1: Erased

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

Bits 30:6 Reserved, must be kept cleared

**Bits 5:0 PCROP1A\_END[5:0]:** PCROP1A area end offset

Contains the offset of the last subpage of the PCROP1A area.

*Note: The number of effective bits depends on the size of the flash memory in the device.*

### 3.7.9 FLASH WRP area A address register (FLASH\_WRP1AR)

Address offset: 0x02C

Reset value: 0x000X 000X (The option bits are loaded with values from flash memory at power-on reset release.)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WRP1A_END[3:0]														
															rw rw rw rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WRP1A_STRT[3:0]														
															rw rw rw rw

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

**Bits 19:16 WRP1A\_END[3:0]:** WRP area A end offset

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

*Note: The number of effective bits depends on the size of the flash memory in the device.*

Bits 15:4 Reserved, must be kept at reset value.

**Bits 3:0 WRP1A\_STRT[3:0]:** WRP area A start offset

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

*Note: The number of effective bits depends on the size of the flash memory in the device.*

### 3.7.10 FLASH WRP area B address register (FLASH\_WRP1BR)

Address offset: 0x030

Reset value: 0x000X 000X (The option bits are loaded with values from flash memory at power-on reset release.)

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	WRP1B_END[3:0]															
														rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	WRP1B_STRT[3:0]															
														rw	rw	rw

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

Bits 19:16 **WRP1B\_END[3:0]**: WRP area B end offset

This bitfield contains the offset of the last page of the WRP area B.

*Note: The number of effective bits depends on the size of the flash memory in the device.*

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **WRP1B\_STRT[3:0]**: WRP area B start offset

This bitfield contains the offset of the first page of the WRP area B.

*Note: The number of effective bits depends on the size of the flash memory in the device.*

### 3.7.11 FLASH PCROP area B start address register (FLASH\_PCROP1BSR)

Address offset: 0x034

Reset value: 0b0000 0000 0000 0000 0000 00XX XXXX (The option bits are loaded with values from flash memory at power-on reset release.)

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

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.	PCROP1B_STRT[5:0]															
														rw	rw	rw

Bits 31:6 Reserved, must be kept cleared

Bits 5:0 **PCROP1B\_STRT[5:0]**: PCROP1B area start offset

Contains the offset of the first subpage of the PCROP1B area.

*Note: The number of effective bits depends on the size of the flash memory in the device.*

### 3.7.12 FLASH PCROP area B end address register (FLASH\_PCROP1BER)

Address offset: 0x038

Reset value: 0b0000 0000 0000 0000 0000 00XX XXXX (The option bits are loaded with values from flash memory at power-on reset release.)

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

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.	PCROP1B_END[5:0]														
											rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept cleared

Bits 5:0 **PCROP1B\_END[5:0]:** PCROP1B area end offset

Contains the offset of the last subpage of the PCROP1B area.

*Note: The number of effective bits depends on the size of the flash memory in the device.*

### 3.7.13 FLASH security register (FLASH\_SECR)

Address offset: 0x080

Reset value: 0b0000 0000 0000 000X 0000 0000 000X XXXX (The option bits are loaded with values from flash memory at power-on reset release.)

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

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

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

Bit 16 **BOOT\_LOCK:** used to force boot from user area

0: Boot based on the pad/option bit configuration

1: Boot forced from Main flash memory

**Caution:** If the bit is set in association with RDP level 1, the debug capabilities are disabled, except in the case of a bad OBL (mismatch).

Bits 15:5 Reserved, must be kept at reset value

Bits 4:0 **SEC\_SIZE[4:0]:** Securable memory area size

Contains the number of securable flash memory pages.

*Note: The number of effective bits depends on the size of the flash memory in the device.*

### 3.7.14 FLASH register map

Table 18. FLASH 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	Res	Res	Res	Res	Res	Res			
	Reset value																																		
0x004	Reserved	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			
0x008	FLASH_KEYR																																		
	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			
0x00C	FLASH_OPTKEYR																																		
	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			
0x010	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	Res	Res	Res	Res			
	Reset value																																		
0x014	FLASH_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																																		
0x018	Reserved																																		
0x020	FLASH_OPTR	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																																		
0x024	FLASH_PCROP1ASR	Res	X	PCROP_RDP	Res																														
	Reset value																																		
0x028	FLASH_PCROP1AER	Res	X	PCROP_RDP	Res																														
	Reset value																																		
0x02C	FLASH_WRP1AR	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																																		
0x030	FLASH_WRP1BR	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																																		
0x034	FLASH_PCROP1BSR	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																																		
0x038	FLASH_PCROP1BER	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																																		
0x03C - 0x07F	Reserved	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			

**Table 18. FLASH register map and reset values (continued)**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BOOT_LOCK	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	FLASH_SECR	Res.	X	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE[4:0]																
	Reset value																											X	X	X	X	X		

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 4 Power control (PWR)

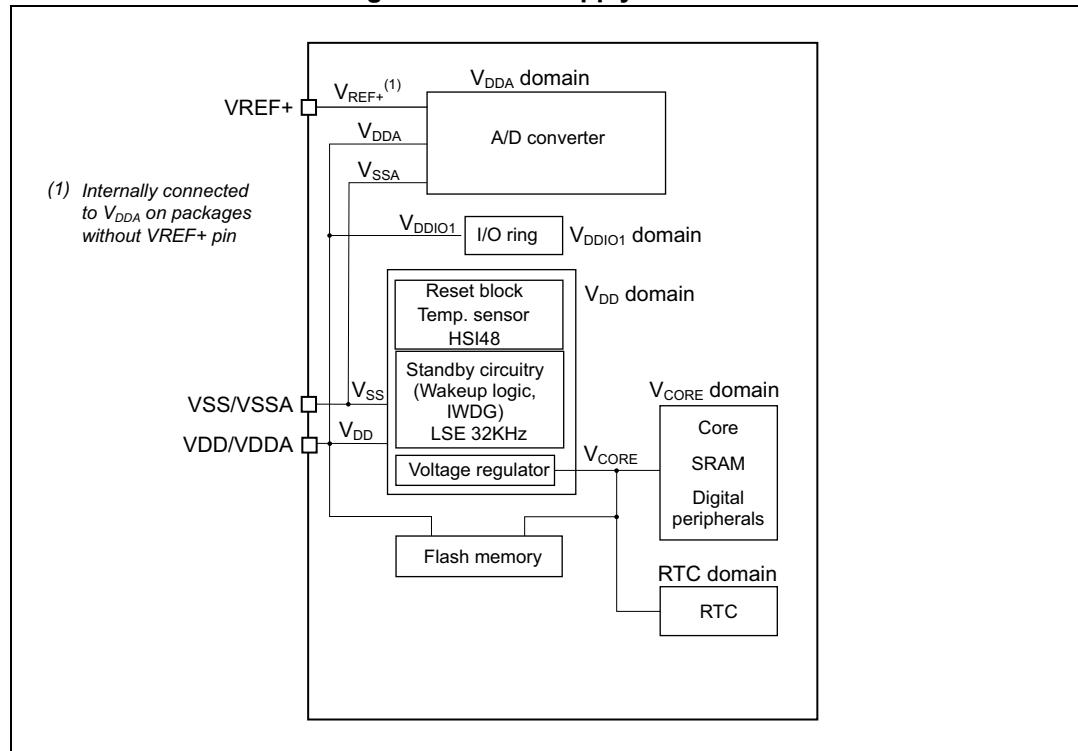
### 4.1 Power supplies and voltage references

The devices are powered through the VDD/VDDA pin. Internally, the power is delivered through the V<sub>DD</sub> power line supplying the internal resources in the V<sub>DD</sub> power domain and the Flash memory, the V<sub>DDA</sub> power line supplying the ADC, and the V<sub>DDIO1</sub> power line supplying the I/O ring.

Through the V<sub>CORE</sub> power line, the voltage regulator supplies the resources in the V<sub>CORE</sub> power domain including the core, SRAM and digital peripherals, the RTC domain and the Flash memory.

Refer to the following figure and to the device datasheets for further information.

**Figure 5. Power supply overview**



#### 4.1.1 ADC reference voltage

V<sub>REF+</sub> defines the full-scale ADC input signal level. On packages with VREF+ input pin, the V<sub>REF+</sub> voltage reference is supplied externally, which allows increasing the ADC resolution (voltage per step) and reducing the noise. Refer to the device datasheets for the allowed V<sub>REF+</sub> range.

#### 4.1.2 Voltage regulator

When enabled, the voltage regulator provides the V<sub>CORE</sub> supply voltage to the V<sub>CORE</sub> domain.

The regulator is enabled upon reset and remains enabled as long as the device operates in Run, Sleep, or Stop mode.

In Standby and Shutdown modes, the regulator is disabled and the  $V_{CORE}$  domain is not powered. As a consequence, the SRAM and register contents are lost.

## 4.2 Power supply supervisor

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

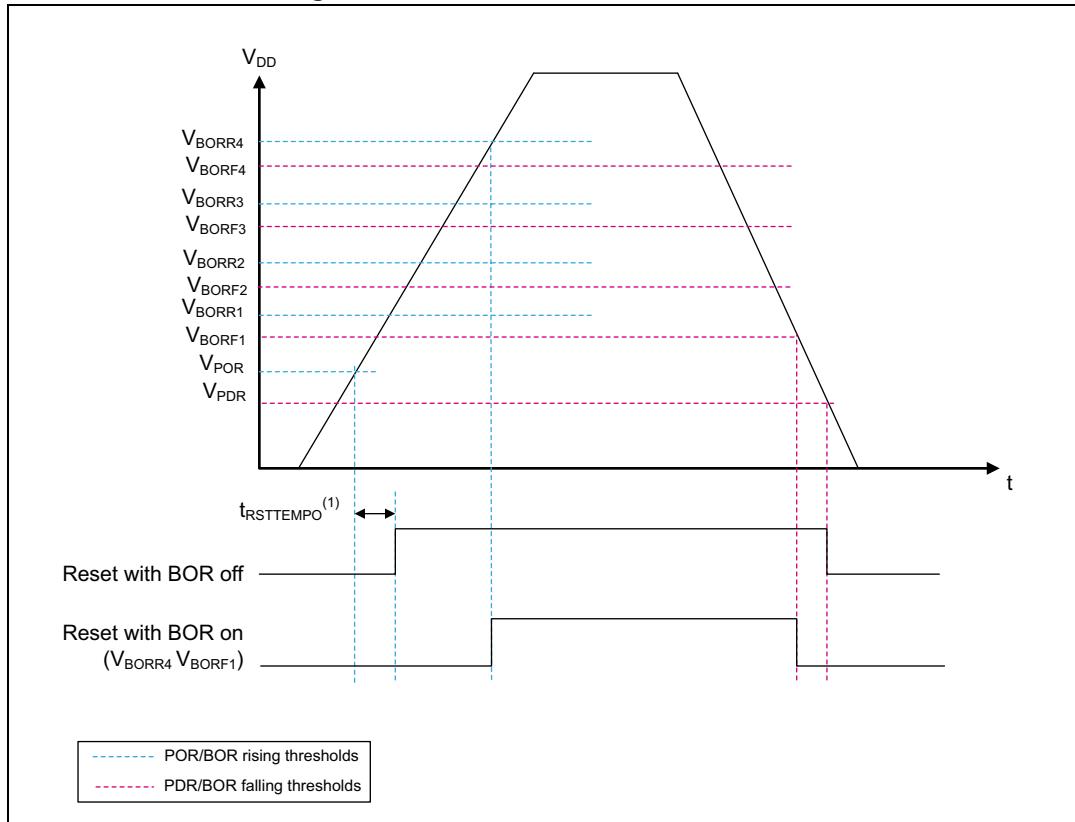
The device features an integrated power-on reset (POR) / power-down reset (PDR), coupled with a brown-out reset (BOR) circuitry. The POR/PDR is active in all power modes. The BOR can be enabled or disabled only through option bytes. It is not available in Shutdown mode.

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

---

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

---

**Figure 6. POR, PDR, and BOR thresholds**

1. The reset temporization  $t_{RSTTEMPO}^{(1)}$  starts when  $V_{DD}$  crosses  $V_{POR}$  threshold, indifferently from the configuration of the BOR Option bits.

For more details on the brown-out reset thresholds, refer to the electrical characteristics section in the datasheet.

## 4.3 Operating modes

The device has a full-operating mode called Run mode and several low-power modes allowing substantial power saving.

Upon a system or a power-on reset, the device starts operating in Run mode.

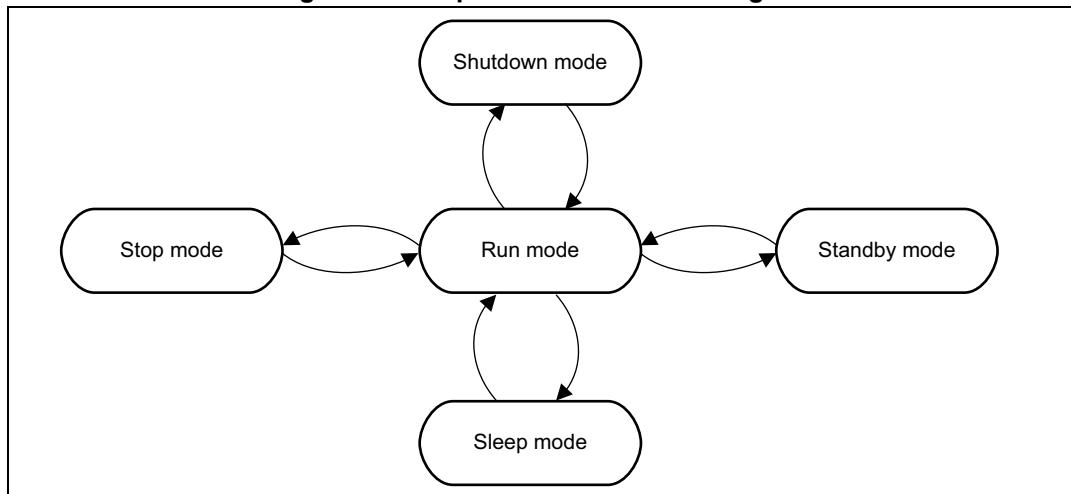
Sleep, Stop, Standby, and Shutdown low-power modes are available to save power when there is no need for the CPU to execute instructions, for example when waiting for an external event.

Different low-power modes offer different trade-offs between power consumption, startup speed and wakeup possibilities. While the Sleep mode offers the highest agility at cost of the least power saving, the Shutdown mode provides the lowest power consumption at cost of slower wakeup and absence of power supply monitoring and BOR/PDR. The Standby mode, compared to Shutdown, keeps the LSI oscillator, the IWDG and the voltage monitoring active, at cost of a slightly greater power consumption. The Stop mode keeps low-speed clocks and peripherals, as well as GPIOs active.

Refer to [Table 19: Device resources enabled in different operating modes](#) and [Table 21: Low-power mode exit overview](#) for all details.

The device can transit from Run mode to any of the low-power modes and from any low-power operating mode to Run mode. Transiting from one low-power mode to another is not possible. Refer to the following figure.

Figure 7. Low-power mode transit diagram



The following table gives an overview of the device resources available in each operating mode and their capability to wake the device up from a low-power mode to Run mode. Refer to the table footnotes for complementary information.

**Table 19. Device resources enabled in different operating modes**

Function	Operating mode <sup>(1)</sup>					
	Run	Sleep	Stop <sup>(2)</sup>	Standby <sup>(2)</sup>	Shutdown <sup>(2)</sup>	
CPU	Y	-	-	-	-	-
Flash memory	Y	Y	A <sup>(3)</sup>	-	-	-
SRAM	Y	Y	U	-	-	-
V <sub>CORE</sub> supply <sup>(4)</sup>	Y	Y	Y	-	-	-
BOR/POR/PDR	O	O	U	U	U	U
NRST	Y	Y	Y	Y	Y	Y
DMA/DMAMUX	O	U	-	-	-	-
HSI48	Y	U <sup>(6)</sup>	- <sup>(6)</sup>	-	-	-
HSE	O	U	-	-	-	-
LSI	O	U	U	-	U	-
LSE	O	U	U	-	-	-
CSS	O	U	-	-	-	-
CSS on LSE	O	U	U	O	-	-
RTC / Auto wakeup	O	U	U	O	-	-
USART1	O	U <sup>(7)</sup>	U <sup>(7)</sup>	O <sup>(7)</sup>	-	-
USART2	O	U <sup>(7)</sup>	-	-	-	-
I2C1	O	U <sup>(8)</sup>	U <sup>(8)</sup>	O <sup>(8)</sup>	-	-
SPI1/I2S1	O	U	-	-	-	-
ADC	O	U	- <sup>(9)</sup>	-	-	-
Temperature sensor	O	U	- <sup>(9)</sup>	-	-	-
TIMx	O	U	-	-	-	-
IWDG	O	U	U	O	U	O
WWDG	O	U	-	-	-	-
SysTick timer	O	U	-	-	-	-
CRC	O	U	-	-	-	-
GPIOs	O	U	U	O	(10)	(11)
Individual peripheral clocks	O	A <sup>(12)</sup>	A <sup>(12)</sup>	-	-	-

1. Y = (yes): resource enabled upon reset and upon wakeup from Stop, Standby, and Shutdown; O = (optional): resource disabled by default, and possible to enable by software; U = (unchanged): resource kept in the same operating state as before low-power mode entry; A = (automatic): resource can be set for automatic disable/power-down upon transitioning to low-power mode; - = resource not available / without wakeup capability (grayed-out columns)

2. The grayed-out column indicates the capability of the resource to wake the device up from a low-power mode.

3. Possibility of automatic power-down.

4. When not available (voltage regulator disabled), the SRAM and register contents are lost.
5. The supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop.
6. When the HSI48 is off, peripherals with wakeup capability, such as I2C1, can wake the HSI48 oscillator up to allow temporary operation to assess the device wakeup condition such as I<sup>2</sup>C address match, then stop HSI48 if the wakeup condition is not met, or wake the device up if it is met. When the HSI48 is restarted by a peripheral with wake-up capability, during the low-power mode it only delivers clock to that peripheral.
7. USART reception is functional in Sleep and Stop mode. It generates a wakeup interrupt on START, address match, or received frame event. The peripheral has the capability to restart the HSI48 oscillator to assess the address match and received frame event device wakeup conditions.
8. I<sup>2</sup>C address detection is functional in Sleep and Stop mode. It generates a wakeup interrupt on address match. The peripheral has the capability to restart the HSI48 oscillator to assess the address match event device wakeup condition.
9. Keeps consuming idle current unless disabled prior to entering Stop mode.
10. I/Os can be configured to keep, in Standby and Shutdown mode, internal pull-up or pull-down, or to float. Refer to PUCRx and PDCRx registers and the APC bit of the PWR\_CR3 register. This configuration is kept when exiting Standby mode but it is lost when exiting Shutdown mode.
11. WKUPx I/Os (up to five, depending on the package), with wakeup from Standby/Shutdown mode capability.
12. Through the RCC\_AHBSMENR and RCC\_APBSMENRx registers, the software can configure the individual peripheral clocks to automatically disable upon entering Sleep and Stop modes.

#### 4.3.1 Power saving in run mode

The power consumption in Run mode can be reduced through selecting a system clock with lower frequency, scaling down the system clock frequency, disabling unused peripherals and/or stopping their clocks (peripheral clock gating).

##### Slowing down system clocks

The SYSSCLK, HCLK, and PCLK clock frequencies can be reduced with prescalers controlled through prescaler registers. Their settings also apply to the Sleep mode.

For more details, refer to [Section 5.4.3: RCC clock configuration register \(RCC\\_CFGR\)](#).

##### Peripheral clock gating

The HCLK and PCLK to individual peripherals and memories can be stopped (clock gating) at any time to reduce the power consumption.

The RCC\_AHBENR and RCC\_APBENRx registers enable or disable individual clocks to peripherals (clock gating).

#### 4.3.2 Low-power modes

The device features the following low-power modes:

- **Sleep mode**

CPU clock is off, all peripherals including Cortex®-M0+ core peripherals such as NVIC and SysTick can run and wake up the CPU when an interrupt or an event occurs.

To further reduce the power consumption in Sleep mode, the peripheral clocks can be disabled prior to executing the WFI or WFE instructions. This can also be done automatically, by configuring RCC\_AHBSMENR and RCC\_APBSMENRx registers. If disabled before entering Sleep mode, HSI48 can be restarted by a peripheral with wakeup capability requiring HSI48.

- **Stop mode**

SRAM and register contents are retained. HSE and HSI48 stop. HSI48 can be

restarted by a peripheral with wakeup capability requiring HSI48.  
The LSI and LSE oscillators can be kept running.

The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

The event of exiting Stop mode enables the HSI48 oscillator and selects HSISYS as system clock.

- **Standby mode**

$V_{CORE}$  domain is powered off and the SRAM and register contents lost, except [PWR control register 3 \(PWR\\_CR3\)](#) and [PWR backup x register \(PWR\\_BKPxR\)](#).

All clocks in the  $V_{CORE}$  domain are stopped and the HSI48 and HSE oscillators disabled. The IWDG and the LSI oscillator can be kept running.

The event of exiting Standby mode enables the HSI48 oscillator, selects HSISYS as system clock and sets its prescaler division factor to four (HSIDIV[2:0] = 010).

- **Shutdown mode**

$V_{CORE}$  domain is powered off and the SRAM and register contents lost.

All oscillators are disabled.

The event of exiting Shutdown mode enables the HSI48 oscillator, selects HSISYS as system clock and sets its prescaler division factor to four (HSIDIV[2:0] = 010).

In this mode, the supply voltage monitoring is disabled and the product behavior is not guaranteed in case of a power voltage drop.

## Debug in low-power modes

By default, the debug connection is lost upon entering Stop, Shutdown, or Standby mode, as the core is no longer clocked.

However, specific settings in the DBGMCU\_CR register allow debugging the software even in low-power modes. For more details, refer to [Section 26.9.1: Debug support for low-power modes](#).

## Low-power mode entry and exit

The software controls low-power mode entry, selection, and exit through:

- SEVONPEND, SLEEPDEEP, and SLEEPONEXIT bits of the Cortex®-M0+ system control register
- LPMS[2:0] bitfield of the [PWR control register 1 \(PWR\\_CR1\)](#)
- WFI (wait for interrupt) and WFE (wait for event) instructions (low-power mode entry stimuli)
- NVIC, EXTI and peripheral pending interrupt and event flags
- configuring return from ISR (interrupt service routine) as low-power mode entry stimulus

### Entering low-power modes

Conditionally, the device enters low-power modes upon one of the following stimuli:

- WFI (wait for interrupt) instruction
- WFE (wait for event) instruction
- return from ISR (when the SLEEPONEXIT bit is high)

The low-power mode entry stimulus occurring while a low-power mode exit condition is met, is ignored (the low-power mode entry is aborted) and the program execution continues.

For low-power modes other than Sleep, the low-power mode entry is delayed (as opposed to aborted) until a potential ongoing Flash memory or APB access is terminated.

The selection of the low-power mode to enter is determined by the SLEEPDEEP and LPMS[2:0] bitfields.

Refer to the following table (including the table footnotes) for details about low-power mode entry conditions. All the conditions in a table row must be met for the corresponding low-power mode entry to occur.

**Table 20. Low-power mode entry overview**

Low-power mode	Low-power mode entry stimulus	Condition <sup>(1)</sup>							
		SLEEPDEEP	SLEEPONEXIT	LPMS[2:0]	Interrupt pending <sup>(2)</sup>	Event pending <sup>(3)</sup>	WUFX bit set	Flash memory access	APB access
Sleep	WFI	0	-	-	Aa	-	-	-	-
	WFE	0	-	-	-	Aa	-	-	-
	Return from ISR	0	1	-	Aa	-	-	-	-
Stop	WFI	1	-	000	Aa	-	-	Ad	Ad
	WFE	1	-	000	-	Aa	-	Ad	Ad
	Return from ISR	1	1	000	Aa	-	-	Ad	Ad
Standby	WFI	1	-	011	Aa	-	Aa	Ad	Ad
	WFE	1	-	011	-	Aa	Aa	Ad	Ad
	Return from ISR	1	1	011	Aa	-	Aa	Ad	Ad
Shutdown	WFI	1	-	1XX	Aa	-	Aa	Ad	Ad
	WFE	1	-	1XX	-	Aa	Aa	Ad	Ad
	Return from ISR	1	1	1XX	Aa	-	Aa	Ad	Ad

1. “-” = don't care or not applicable. “X” = don't care (bit value). “Aa” = Absent/none; abort the low-power mode entry if present. “Ad” = Absent/none; delay the low-power mode entry as long as present.

2. Any EXTI line interrupt flag (in *EXTI rising edge pending register 1 (EXTI\_RPR1)* and *EXTI falling edge pending register 1 (EXTI\_FPR1)*) or any peripheral wakeup interrupt flag is set.
3. Any enabled EXTI event. For the sleep mode, also any peripheral event with the associated interrupt enabled in the peripheral.

### Exiting low-power modes

The device exits any low-power mode upon external reset on NRST pin.

Additionally, it exits Sleep, Stop, and Standby modes upon BOR/PDR and IWDG reset, and Sleep and Stop modes upon a CSS detection on LSE.

For all other low-power mode exit conditions, refer to the following table (including the table footnotes). All conditions in a table row must be met for the corresponding low-power mode exit to occur.

Table 21. Low-power mode exit overview

Low-power mode	Mode entry stimulus	Condition <sup>(1)</sup>						Wakeup latency	Upon exit		
		SEVONPEND	Peripheral event / Interrupt	EXTI interrupt <sup>(2)</sup>	EXTI event <sup>(3)</sup>	NVIC IRQ interrupt <sup>(4)</sup>	WUFx bit		Clock	HSIDIV [2:0] <sup>(5)</sup>	SBF <sup>(6)</sup>
Sleep	WFI or return from ISR	-	E	-	-	R	-	None	As before entry	As before entry	0
		-	E	R	-	E	-				
	WFE	0	E	-	-	R	-				
		0	E	R	-	E	-				
		0	E	-	R	-	-				
		1	E	-	-	R	-				
		1	E	R	-	-	-				
		1	E	-	R	-	-				
Stop	WFI or return from ISR	-	E	R	-	E	-	HSI48 / Flash memory startup <sup>(7)</sup>	HSISYS	As before entry	0
		-	E	R	-	E	-				
	WFE	0	E	R	-	E	-				
		0	E	-	R	-	-				
		1	E	R	-	-	-				
Standby	-	-	-	-	-	-	R	Reset phase	HSISYS	010	1
Shutdown	-	-	-	-	-	-	R	Reset phase	HSISYS	010	0

1. “-” = don't care or not applicable, “E” = enabled, “R” = enabled and raised.

2. Any EXTI line set as unmasked in [EXTI CPU wakeup with interrupt mask register \(EXTI\\_IMR1\)](#).

3. Any EXTI line set as unmasked in [EXTI CPU wakeup with event mask register \(EXTI\\_EMR1\)](#)

4. Any IRQ interrupt listed in [Table 40: Vector table](#) and activated in the NVIC\_ISER register (refer to the product programming manual).

5. Bitfield of [RCC clock control register \(RCC\\_CR\)](#), controlling the HSISYS prescale. The value HSIDIV[2:0] = 010 corresponds to division by four.

6. Flag in [PWR status register 1 \(PWR\\_SR1\)](#).

7. The longer of HSI48 oscillator startup time and Flash memory startup time (from Stop mode).

- Note:** For any NVIC IRQ interrupt, EXTI interrupt or EXTI event, the table assumes that the mechanism is activated through the NVIC ISER, EXTI\_IMR1, or EXTI\_EMR1 register, respectively. For any interrupt or event, it also has to be activated in the peripheral.
- Some peripherals send their interrupt request signal to both NVIC and EXTI and can generate both NVIC IRQ and EXTI interrupts if configured to do so. Some other peripherals only send their interrupt signal to NVIC and they can therefore only generate an NVIC interrupt. The latter type of peripheral cannot wake the system up from Stop mode.
- Upon waking up from Sleep and Stop modes, the pending bits associated to the interrupt/event having woken up the system must be cleared. It may also be necessary to clear the interrupt flag in the peripheral.
- Upon waking up from Standby and Shutdown mode, the program execution restarts in the same way as upon a reset (boot pin sampling, option bytes loading, reset vector is fetched, and so on).

#### Auto-wakeup from Stop mode

The RTC can wake the device up from Stop mode at regular intervals, without any external stimulus. For this purpose, select LSI or LSE as RTC clock source, through the RTCSEL[1:0] bitfield of the [RCC control/status register 1 \(RCC\\_CSR1\)](#).

The LSI oscillator does not require an external quartz and reduces the system cost, at expense of accuracy. The LSE oscillator with an external quartz ensures higher accuracy but it leads to an extra cost.

To enable the wakeup from Stop mode with RTC alarm:

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

## 4.4 PWR registers

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

### 4.4.1 PWR control register 1 (PWR\_CR1)

The register is reset after wakeup from Standby mode.

Address offset: 0x00

Reset value: 0x0000 0208

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.	FPD_SLP	Res.	FPD_STOP	LPMS[2:0]											
										rw		rw	rw	rw	rw

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

Bit 5 **FPD\_SLP**: Flash memory powered down during Sleep mode

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

0: Flash memory idle

1: Flash memory powered down

Bit 4 Reserved, must be kept at reset value.

Bit 3 **FPD\_STOP**: Flash memory powered down during Stop mode

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

0: Flash memory idle

1: Flash memory powered down

Bits 2:0 **LPMS[2:0]**: Low-power mode selection

These bits select the low-power mode entered when CPU enters deepsleep mode.

000: Stop mode

001: Reserved

010: Reserved

011: Standby mode

1XX: Shutdown mode

#### 4.4.2 PWR control register 3 (PWR\_CR3)

The register is not reset when exiting Standby mode and with the PWRRST bit of the *RCC APB peripheral reset register 1 (RCC\_APBRSTR1)*.

Address offset: 0x08

Reset value: 0x0000 8000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIWUL	Res.	Res.	Res.	Res.	APC	Res.	Res.	Res.	Res.	EWUP 6	Res.	EWUP 4	EWUP 3	EWUP 2	EWUP 1
rw					rw					rw		rw	rw	rw	rw

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

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

When set, a rising edge on the internal wakeup line triggers a wakeup event.

0: Disable

1: Enable

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

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

This bit determines whether the I/O pull-up and pull-down configurations defined in the PWR\_PUCRx and PWR\_PDCRx registers are applied.

0: Not applied

1: Applied

Bits 9:6 Reserved, must be kept at reset value.

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

When this bit is set, the WKUP6 external wakeup pin is enabled and triggers a wakeup event when a rising or a falling edge occurs. The active edge is configured through WP6 bit in the PWR\_CR4 register.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **EWUP4**: Enable WKUP4 wakeup pin

When this bit is set, the WKUP4 external wakeup pin is enabled and triggers a wakeup event when a rising or a falling edge occurs. The active edge is configured via the WP4 bit in the PWR\_CR4 register.

Bit 2 **EWUP3**: Enable WKUP3 wakeup pin

When this bit is set, the WKUP3 external wakeup pin is enabled and triggers a wakeup event when a rising or a falling edge occurs. The active edge is configured via the WP3 bit of the PWR\_CR4 register.

Bit 1 **EWUP2**: Enable WKUP2 wakeup pin

When this bit is set, the WKUP2 external wakeup pin is enabled and triggers a wakeup event when a rising or a falling edge occurs. The active edge is configured via the WP2 bit of the PWR\_CR4 register.

Bit 0 **EWUP1**: Enable WKUP1 wakeup pin

When this bit is set, the WKUP1 external wakeup pin is enabled and triggers a wakeup event when a rising or a falling edge occurs. The active edge is configured via the WP1 bit of the PWR\_CR4 register.

#### 4.4.3 PWR control register 4 (PWR\_CR4)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x0C

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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.	WP6	Res.	WP4	WP3	WP2	WP1									
										rw		rw	rw	rw	rw

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

Bit 5 **WP6**: WKUP6 wakeup pin polarity

WKUP6 external wakeup signal polarity (level or edge) to generate wakeup condition:

- 0: High level or rising edge
- 1: Low level or falling edge

Bit 4 Reserved, must be kept at reset value.

Bit 3 **WP4**: WKUP4 wakeup pin polarity

WKUP4 external wakeup signal polarity (level or edge) to generate wakeup condition:

- 0: High level or rising edge
- 1: Low level or falling edge

Bit 2 **WP3**: WKUP3 wakeup pin polarity

WKUP3 external wakeup signal polarity (level or edge) to generate wakeup condition:

- 0: High level or rising edge
- 1: Low level or falling edge

Bit 1 **WP2**: WKUP2 wakeup pin polarity

WKUP2 external wakeup signal polarity (level or edge) to generate wakeup condition:

- 0: High level or rising edge
- 1: Low level or falling edge

Bit 0 **WP1**: WKUP1 wakeup pin polarity

WKUP1 external wakeup signal polarity (level or edge) to generate wakeup condition:

- 0: High level or rising edge
- 1: Low level or falling edge

#### 4.4.4 PWR status register 1 (PWR\_SR1)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x10

Reset value: 0x0000 0000

Access: Requires two extra APB clock cycles, compared to standard APB access.

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
WUF1	Res.	Res.	Res.	Res.	Res.	Res.	SBF	Res.	Res.	WUF6	Res.	WUF4	WUF3	WUF2	WUF1
r							r			r		r	r	r	r

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

Bit 15 **WUF1**: Wakeup flag internal

This bit is set when a wakeup condition is detected on the internal wakeup line. It is cleared when all internal wakeup sources are cleared.

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

Bit 8 **SBF**: Standby flag

This bit is set by hardware when the device enters Standby mode and is cleared by setting the CSBF bit in the PWR\_SCR register, or by a power-on reset. It is not cleared by the system reset.

0: The device did not enter Standby mode

1: The device entered Standby mode

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

Bit 5 **WUF6**: Wakeup flag 6

This bit is set when a wakeup condition is detected on WKUP6 wakeup pin. It is cleared by setting the CWUF6 bit of the PWR\_SCR register.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **WUF4**: Wakeup flag 4

This bit is set when a wakeup condition is detected on WKUP4 wakeup pin. It is cleared by setting the CWUF4 bit of the PWR\_SCR register.

Bit 2 **WUF3**: Wakeup flag 3

This bit is set when a wakeup condition is detected on WKUP3 wakeup pin. It is cleared by setting the CWUF3 bit of the PWR\_SCR register.

Bit 1 **WUF2**: Wakeup flag 2

This bit is set when a wakeup condition is detected on WKUP2 wakeup pin. It is cleared by setting the CWUF2 bit of the PWR\_SCR register.

Bit 0 **WUF1**: Wakeup flag 1

This bit is set when a wakeup condition is detected on WKUP1 wakeup pin. It is cleared by setting the CWUF1 bit of the PWR\_SCR register.

#### 4.4.5 PWR status register 2 (PWR\_SR2)

This register is reset when exiting Standby/Shutdown modes.

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.								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FLASH_RDY	Res.													
								r							

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

**Bit 7 FLASH\_RDY:** Flash ready flag

This bit is set by hardware to indicate when the Flash memory is ready to be accessed after wakeup from power-down. To place the Flash memory in power-down, set either FPD\_SLP or FPD\_STP bit.

0: Flash memory in power-down

1: Flash memory ready to be accessed

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

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

#### 4.4.6 PWR status clear register (PWR\_SCR)

Address offset: 0x18

Reset value: 0x0000 0000

Access: Requires three extra APB clock cycles, compared to standard APB access.

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.	CSBF	Res.	Res.	CWUF 6	Res.	CWUF 4	CWUF 3	CWUF 2	CWUF 1						
							w			w		w	w	w	w

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

**Bit 8 CSBF:** Clear standby flag

Setting this bit clears the SBF flag in the PWR\_SR1 register.

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

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

Setting this bit clears the WUF6 flag in the PWR\_SR1 register.

Bit 4 Reserved, must be kept at reset value.

**Bit 3 CWUF4:** Clear wakeup flag 4

Setting this bit clears the WUF4 flag in the PWR\_SR1 register.

**Bit 2 CWUF3:** Clear wakeup flag 3

Setting this bit clears the WUF3 flag in the PWR\_SR1 register.

**Bit 1 CWUF2:** Clear wakeup flag 2

Setting this bit clears the WUF2 flag in the PWR\_SR1 register.

**Bit 0 CWUF1:** Clear wakeup flag 1

Setting this bit clears the WUF1 flag in the PWR\_SR1 register.

#### 4.4.7 PWR Port A pull-up control register (PWR\_PUCRA)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x20

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw															

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

Bits 15:0 **PUi**: Port A pull-up bit i (i = 15 to 0)

Setting PU*i* bit while the corresponding PDi bit is zero and the APC bit of the PWR\_CR3 register is set activates a pull-up device on the PA[i] I/O.

#### 4.4.8 PWR Port A pull-down control register (PWR\_PDCRA)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x24

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw															

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

Bits 15:0 **PDi**: Port A pull-down bit i (i = 15 to 0)

Setting PDi bit while the APC bit of the PWR\_CR3 register is set activates a pull-down device on the PA[i] I/O.

#### 4.4.9 PWR Port B pull-up control register (PWR\_PUCRB)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x28

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw															

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

Bits 15:0 **PUi**: Port B pull-up bit i (i = 15 to 0)

Setting PU*i* bit while the corresponding PD*i* bit is zero and the APC bit of the PWR\_CR3 register is set activates a pull-up device on the PB[i] I/O.

Note: On STM32C011xx, only PU7 and PU6 are available

#### 4.4.10 PWR Port B pull-down control register (PWR\_PDCRB)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x2C

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw															

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

Bits 15:0 **PD*i***: Port B pull-down bit i (i = 15 to 0)

Setting PD*i* bit while the APC bit of the PWR\_CR3 register is set activates a pull-down device on the PB[i] I/O.

Note: On STM32C011xx, only PD7 and PD6 are available

#### 4.4.11 PWR Port C pull-up control register (PWR\_PUCRC)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x30

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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
PU15	PU14	PU13	Res.	Res.	Res.	Res.	Res.	PU7	PU6	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw						rw	rw						

Bits 31:16, 12:8, 5:0 Reserved, must be kept at reset value.

Bits 15:13,7:6 **PUi**: Port C pull-up bit i (i = 15 to 13, 7 to 6)

Setting PU*i* bit while the corresponding PDi bit is zero and the APC bit of the PWR\_CR3 register is set activates a pull-up device on the PC[i] I/O.

Note: On STM32C011xx, only PU15 and PU14 are available

#### 4.4.12 PWR Port C pull-down control register (PWR\_PDCRC)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x34

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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
PD15	PD14	PD13	Res.	Res.	Res.	Res.	Res.	PD7	PD6	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw						rw	rw						

Bits 31:16, 12:8, 5:0 Reserved, must be kept at reset value.

Bits 15:13,7:6 **PDi**: Port C pull-down bit i (i = 15, 14, 13, 7, 6)

Setting PD*i* bit while the APC bit of the PWR\_CR3 register is set activates a pull-down device on the PC[i] I/O.

Note: On STM32C011xx, only PD15 and PD14 are available.

#### 4.4.13 PWR Port D pull-up control register (PWR\_PUCRD)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x38

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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.	PU3	PU2	PU1	PU0											
												rw	rw	rw	rw

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

Bits 3:0 **PUi**: Port D pull-up bit i (i = 3 to 0)

Setting PU<sub>i</sub> bit while the corresponding PD<sub>i</sub> bit is zero and the APC bit of the PWR\_CR3 register is set activates a pull-up device on the PD[i] I/O.

Note: Not available on STM32C011xx.

#### 4.4.14 PWR Port D pull-down control register (PWR\_PDCRD)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x3C

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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.	PD3	PD2	PD1	PD0											
												rw	rw	rw	rw

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

Bits 3:0 **PD<sub>i</sub>**: Port D pull-down bit i (i = 3 to 0)

Setting PD<sub>i</sub> bit while the APC bit of the PWR\_CR3 register is set activates a pull-down device on the PD[i] I/O.

Note: Not available on STM32C011xx.

#### 4.4.15 PWR Port F pull-up control register (PWR\_PUCRF)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x48

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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.	PU2	PU1	PU0												
													rw	rw	rw

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

Bits 2:0 **PUi**: Port F pull-up bit i (i = 2 to 0)

Setting PU $i$  bit while the corresponding PD $i$  bit is zero and the APC bit of the PWR\_CR3 register is set activates a pull-up device on the PF[ $i$ ] I/O.

Note: On STM32C011xx, only PU2 is available.

#### 4.4.16 PWR Port F pull-down control register (PWR\_PDCRF)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x4C

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB access.

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.	PD2	PD1	PD0												
													rw	rw	rw

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

Bits 2:0 **PD $i$** : Port F pull-down bit i (i = 2 to 0)

Setting PD $i$  bit while the APC bit of the PWR\_CR3 register is set activates a pull-down device on the PF[ $i$ ] I/O.

Note: On STM32C011xx, only PD2 is available.

#### 4.4.17 PWR backup x register (PWR\_BKPxR)

The register is not reset when exiting Standby mode and with the PWRRST bit of the [RCC APB peripheral reset register 1 \(RCC\\_APBRSTR1\)](#).

Address offset: 0x070 + 0x04 \* x, (x = 0 to 3)

Reset value: 0x0000 0000

Access: Requires three (writing) or two (reading) extra APB clock cycles, compared to standard APB 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
BKP[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 **BKP[15:0]**: Backup bitfield

This bitfield retains information when the device is in Standby.

#### 4.4.18 PWR register map

Table 22. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x000	PWR_CR1	Res.															
	Reset value																
0x004	Reserved	Res.															
0x008	PWR_CR3	Res.															
	Reset value																
0x00C	PWR_CR4	Res.															
	Reset value																
0x010	PWR_SR1	Res.															
	Reset value	0	WUF1	Res.													
0x014	PWR_SR2	Res.															
	Reset value	0	SBF	Res.													
○ FLASH_RDY																	
○ FPD_SLP																	
○ FPD_STOP																	
○ LPMS [2:0]																	

**Table 22. PWR register map and reset values (continued)**

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 5 Reset and clock control (RCC)

### 5.1 Reset

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

#### 5.1.1 Power reset

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

- power-on reset (POR) or brown-out reset (BOR)
- exit from Standby mode
- exit from Shutdown mode

Power and brown-out reset set all registers to their reset values.

When exiting Standby mode, all registers in the V<sub>CORE</sub> domain are set to their reset value. Registers outside the V<sub>CORE</sub> domain (WKUP, IWDG, and Standby/Shutdown mode control) are not impacted.

When exiting Shutdown mode, the brown-out reset is generated, resetting all registers.

#### 5.1.2 System reset

System reset sets all registers to their reset values except the reset flags in the RCC control/status register 2 (RCC\_CSR2) and the registers in the RTC domain.

System reset is generated when one of the following events occurs:

- low level on the NRST pin (external reset)
- window watchdog event (WWDG reset)
- independent watchdog event (IWDG reset)
- software reset (SW reset) (see [Software reset](#))
- low-power mode security reset (see [Low-power mode security reset](#))
- option byte loader reset (see [Option byte loader reset](#))
- power-on reset

The reset source can be identified by checking the reset flags in the RCC\_CSR register (see [Section 5.4.21: RCC control/status register 2 \(RCC\\_CSR2\)](#)).

#### NRST pin (external reset):

Through specific option bits, the NRST pin is configurable for operating as:

- **Reset input/output** (default at device delivery)

Valid reset signal on the pin is propagated to the internal logic, and each internal reset source is led to a pulse generator the output of which drives this pin. The GPIO functionality (PF2) is not available. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source to be output on the NRST pin. An internal reset holder option can be used, if enabled in the [FLASH option register \(FLASH\\_OPTR\)](#), to ensure that the pin is pulled low until its voltage meets V<sub>IL</sub>

threshold. This function allows the detection of internal reset sources by external components when the line faces a significant capacitive load.

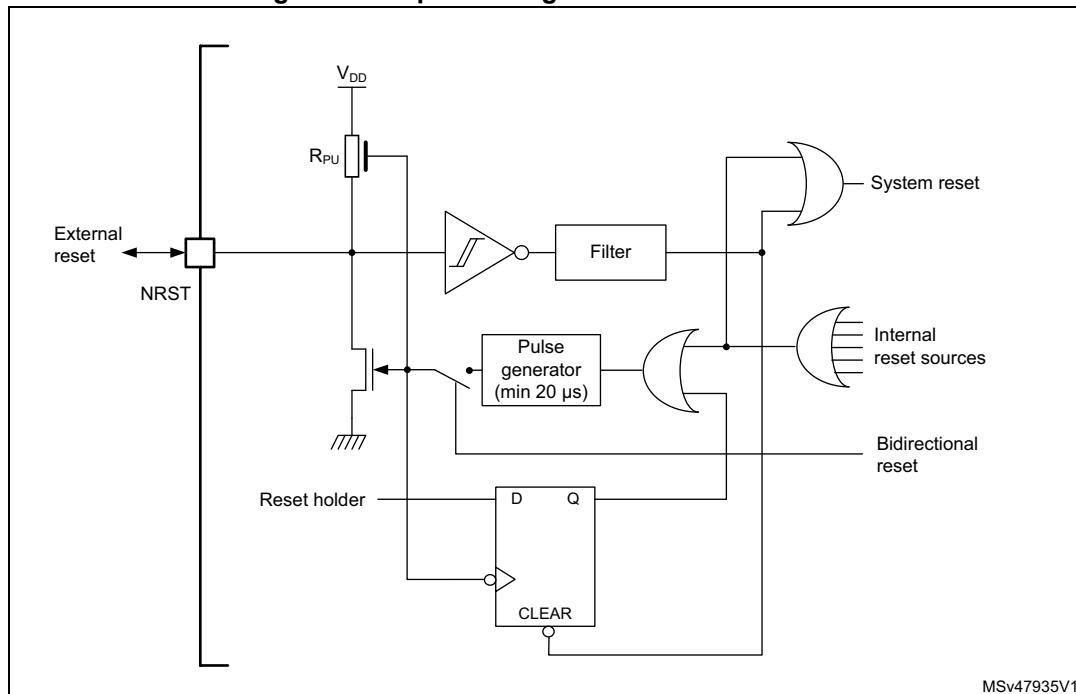
- **Reset input**

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

- **GPIO**

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

**Figure 8. Simplified diagram of the reset circuit**



**Caution:** Upon power reset or wakeup from shutdown mode, the NRST pin is configured as Reset input/output and driven low by the system until it is reconfigured to the expected mode when the option bytes are loaded, in the fourth clock cycle after the end of  $t_{RSTTEMPO}$  time (see datasheet).

### Software reset

The SYSRESETREQ bit in Cortex®-M0+ Application interrupt and reset control register must be set to force a software reset on the device (refer to the programming manual PM0223).

### Low-power mode security reset

To prevent that critical applications mistakenly enter a low-power mode, low-power mode security resets are available. If enabled in option bytes, the resets are generated in the following conditions:

- **Entering Standby mode**

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

- **Entering Stop mode**

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

- **Entering Shutdown mode**

This type of reset is enabled by resetting nRST\_SHDW bit in user option bytes. In this case, whenever a Shutdown mode entry sequence is successfully executed, the device is reset instead of entering Shutdown mode.

For further information on the user option bytes, refer to [Section 3.4.1: FLASH option byte description](#).

### Option byte loader reset

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

## 5.1.3 RTC domain reset

The RTC domain has two specific resets.

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

- **Software reset**, triggered by setting the RTCRST bit of the [RCC control/status register 1 \(RCC\\_CSR1\)](#).
- **V<sub>DD</sub> power on**.

RTC domain reset only affects the LSE oscillator, the RTC, and the RCC control/status register 1.

## 5.2 Clocks

The device provides the following clock sources producing primary clocks:

- **HSI48 RC** - a high-speed fully-integrated RC oscillator producing **HSI48** clock (48 MHz)
- **HSE OSC** - a high-speed oscillator with external crystal/ceramic resonator or external clock source, producing **HSE** clock (4 to 48 MHz)
- **LSI RC** - a low-speed fully-integrated RC oscillator producing **LSI** clock (about 32 kHz)
- **LSE OSC** - a low-speed oscillator with external crystal/ceramic resonator or external clock source, producing **LSE** clock (accurate 32.768 kHz or external clock up to 1 MHz)
- **I2S\_CKIN** - pin for direct clock input for I2S1 peripheral

Each oscillator can be switched on or off independently when it is not used, to optimize power consumption. Check sub-sections of this section for more functional details. For electrical characteristics of the internal and external clock sources, refer to the device datasheet.

The device produces secondary clocks by dividing the primary clocks:

- **HSISYS** - a clock derived from HSI48 through division by a factor programmable from 1 to 128
- **SYSCLK** - a clock obtained through selecting one of LSE, LSI, HSE and HSISYS clocks
- **HSIKER** - a clock derived from HSI48 through division by a factor programmable from 1 to 8
- **HCLK** - a clock derived from SYSCLK through division by a factor programmable from 1 to 512
- **HCLK8** - a clock derived from HCLK through division by eight
- **PCLK** - a clock derived from HCLK through division by a factor programmable from 1 to 16
- **TIMPCLK** - a clock derived from PCLK, running at PCLK frequency if the APB prescaler division factor is set to 1, or at twice the PCLK frequency otherwise

More secondary clocks are generated by fixed division of HSE, HSI48 and HCLK clocks.

The HSISYS is used as system clock source after startup from reset, with the division by four (producing 12 MHz frequency).

The HCLK clock and PCLK clock are used for clocking the AHB and the APB domains, respectively. Their maximum allowed frequency is 48 MHz.

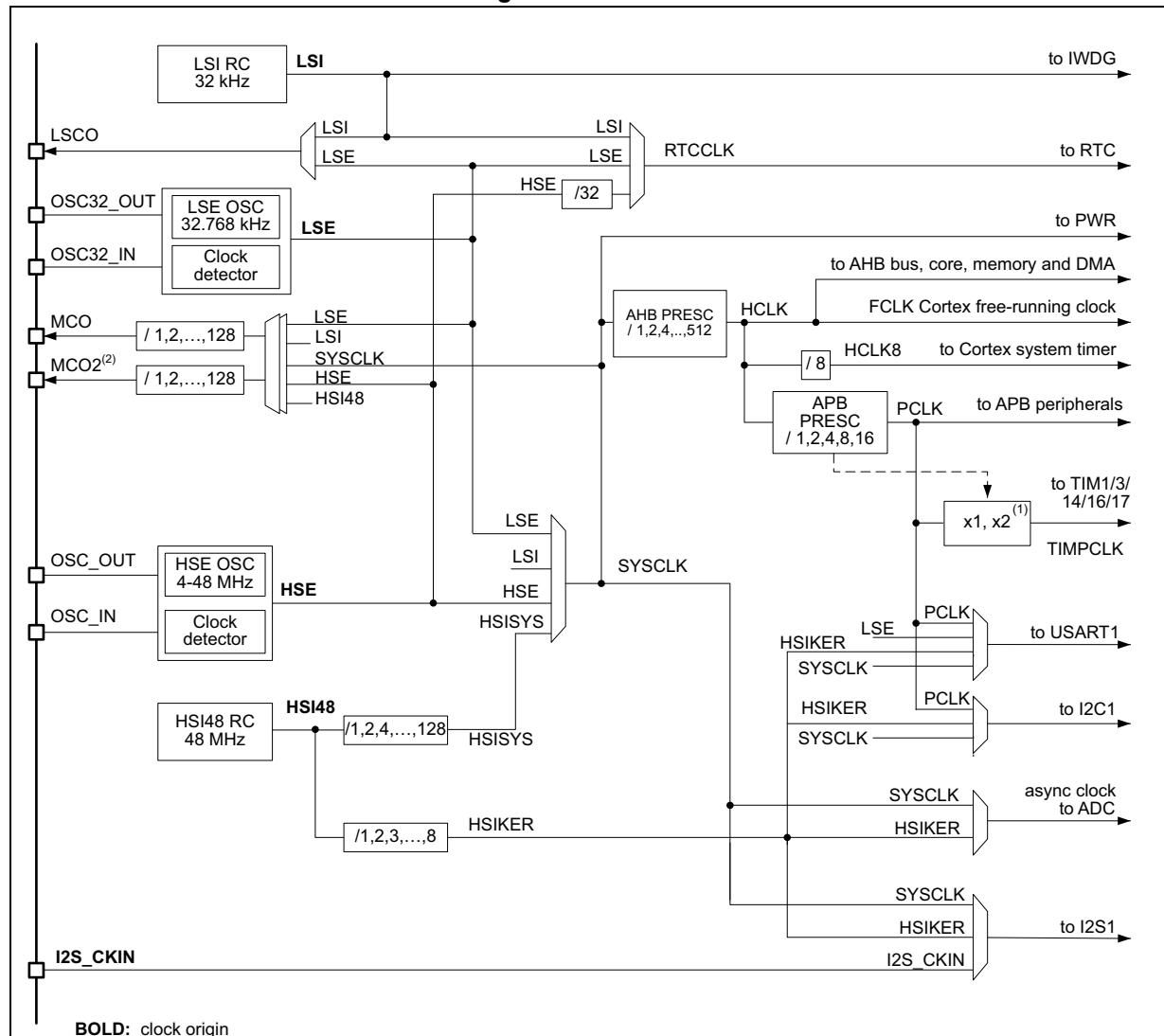
The peripherals are clocked with the clocks from the bus they are attached to (HCLK for AHB, PCLK for APB) except:

- **TIMx**
    - TIMPCLK running at PCLK frequency if the APB prescaler division factor is set to 1, or at twice the PCLK frequency otherwise
  - **USART1**, with these clock sources to select from:
    - SYSCLK (system clock)
    - HSIKER
    - LSE
    - PCLK (APB clock)
- The wakeup from Stop mode is supported only when the clock is HSI48 or LSE.
- **ADC**, with these clock sources to select from:
    - SYSCLK (system clock)
    - HSIKER

- **I2Cx**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSIKER
  - PCLK (APB clock)The wakeup from Stop mode is supported only when the clock is HSI48.
- **I2Sx**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSIKER
  - I2S\_CKIN pin
- **RTC**, with these clock sources to select from:
  - LSE
  - LSI
  - HSE clock divided by 32The functionality in Stop mode (including wakeup) is supported only when the clock is LSI or LSE.
- **IWDG**, always clocked with LSI clock.
- **SysTick** (Cortex® core system timer), with these clock sources to select from:
  - HCLK (AHB clock)
  - HCLK clock divided by 8The selection is done through SysTick control and status register.

HCLK is used as Cortex®-M0+ free-running clock (FCLK). For more details, refer to the programming manual PM0223.

Figure 9. Clock tree



1. TIMPCLK runs at PCLK frequency if the APB prescaler division factor is set to 1, or at twice the PCLK frequency otherwise

### 5.2.1 HSE clock

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

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

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

**Figure 10. HSE/ LSE clock sources**

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	

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

The 4 to 48 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

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

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

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

### External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 48 MHz. This mode is selected by setting the HSEBYP and HSEON bits in the [\*RCC clock control register \(RCC\\_CR\)\*](#). The external clock signal (square, sinus or triangle - refer to the *datasheet*) must drive the OSC\_IN pin, on devices where OSC\_IN and OSC\_OUT pins are available (see [Figure 10](#)). The OSC\_OUT pin can be used as a GPIO.

The OSC\_OUT pin can be used as a GPIO or it can be configured as OSC\_EN alternate function, to provide an enable signal to external clock synthesizer. It allows stopping the external clock source when the device enters low power modes.

**Note:** For details on pin availability, refer to the pinout section in the corresponding device datasheet.

To minimize the consumption, it is recommended to use the square signal.

## 5.2.2 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator.

The HSI48 RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator. However, even after calibration, it is less accurate than an oscillator using a frequency reference such as quartz crystal or ceramic resonator.

The HSISYS clock derived from HSI48 can be selected as system clock after wakeup from Stop mode. Refer to [Section 5.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 5.2.6: Clock security system \(CSS\)](#).

### Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. To compensate for this variation, each device is factory calibrated to 1 % accuracy at  $T_A=25^\circ\text{C}$ .

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the [RCC internal clock source calibration register \(RCC\\_ICSCR\)](#).

Voltage or temperature variations in the application may affect the HSI48 frequency of the RC oscillator. It can be trimmed using the HSITRIM[6:0] bits in the [RCC internal clock source calibration register \(RCC\\_ICSCR\)](#).

For more details on how to measure the HSI48 frequency variation, refer to [Section 5.2.13: Internal/external clock measurement with TIM14/TIM16/TIM17](#).

The HSIRDY flag in the [RCC clock control register \(RCC\\_CR\)](#) indicates if the HSI48 RC is stable or not. At startup, the HSI48 RC output clock is not released until this bit is set by hardware.

The HSI48 RC can be switched on and off using the HSION bit in the [RCC clock control register \(RCC\\_CR\)](#).

The HSI48 signal can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 5.2.6: Clock security system \(CSS\) on page 111](#).

## 5.2.3 LSE clock

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

The LSE crystal is switched on and off using the LSEON bit of [RCC control/status register 1 \(RCC\\_CSR1\)](#). The crystal oscillator driving strength can be changed at runtime using the LSEDRV bit of the [RCC control/status register 1 \(RCC\\_CSR1\)](#) to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV cleared) when the LSE is ON. However, once LSEDRV is selected, the drive capability can not be increased if LSEON is set.

The LSERDY flag in the [RCC control/status register 1 \(RCC\\_CSR1\)](#) indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the [RCC clock interrupt enable register \(RCC\\_CIER\)](#).

#### External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the [RCC AHB peripheral clock enable in Sleep/Stop mode register \(RCC\\_AHBSMENR\)](#). The external clock signal (square, sinus or triangle - refer to the datasheet) has to drive the OSCX\_IN pin while the OSCX\_OUT pin can be used as GPIO. See [Figure 10](#).

### 5.2.4 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and RTC. The clock frequency is 32 kHz. For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the [RCC control/status register 2 \(RCC\\_CSR2\)](#).

The LSIRDY flag in the [RCC control/status register 2 \(RCC\\_CSR2\)](#) 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 [RCC clock interrupt enable register \(RCC\\_CIER\)](#).

### 5.2.5 System clock (SYSCLK) selection

One of the following clocks can be selected as system clock (SYSCLK):

- LSI
- LSE
- HSISYS
- HSE

The system clock maximum frequency is 48 MHz. Upon system reset, the HSISYS clock derived from HSI48 oscillator is selected as system clock. When a clock source is used 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). If a clock source which is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the [RCC internal clock source calibration register \(RCC\\_ICSCR\)](#) indicate which clock(s) is (are) ready and which clock is currently used as a system clock.

### 5.2.6 Clock security system (CSS)

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

If a failure is detected on the HSE clock:

- the HSE oscillator is automatically disabled
- a clock failure event is sent to the break input of TIM1, TIM16 and TIM17 timers
- CSSI (clock security system interrupt) is generated

The CSSI is linked to the Cortex®-M0+ NMI (non-maskable interrupt) exception vector. It makes the software aware of a HSE clock failure to allow it to perform rescue operations.

**Note:** *If the CSS is enabled and the HSE clock fails, the CSSI occurs and an NMI is automatically generated. The NMI is executed infinitely unless the CSS interrupt pending bit is cleared. It is therefore necessary that the NMI ISR clears the CSSI by setting the CSSC bit in the [RCC clock interrupt clear register \(RCC\\_CICR\)](#).*

If HSE is selected directly or indirectly as system clock, and a failure of HSE clock is detected, the system clock switches automatically to HSISYS and the HSE oscillator is disabled.

### 5.2.7 Clock security system for LSE clock (LSECSS)

A clock security system on LSE can be activated by setting the LSECSSON bit in [RCC control/status register 1 \(RCC\\_CSR1\)](#). This bit can be cleared only by a hardware reset or RTC software reset, or after LSE clock failure detection. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSION enabled) and ready (LSERDY and LSIRDY flags set by hardware), and after selecting the RTC clock by RTCSEL.

The LSECSS works in all modes except Standby and Shutdown. It keeps working also under system reset (excluding power-on reset). If a failure is detected on the LSE oscillator, the LSE clock is no longer supplied to the RTC but its registers are not impacted.

**Note:** *If the LSECSS is enabled and the LSE clock fails, the LSECSSI occurs and an NMI is automatically generated. The NMI is executed infinitely unless the LSECSS interrupt pending bit is cleared. It is therefore necessary that the NMI ISR clears the LSECSSI by setting the LSECSSC bit in the [RCC clock interrupt clear register \(RCC\\_CICR\)](#).*

If LSE is used as system clock, and a failure of LSE clock is detected, the system clock switches automatically to LSI. In low-power modes, an LSE clock failure generates a wakeup. The interrupt flag must then be cleared within the RCC registers.

The software **must** then disable the LSECSSON bit, stop the defective 32 kHz oscillator (by clearing LSEON), and change the RTC clock source (no clock, LSI or HSE, with RTCSEL), or take any appropriate action to secure the application.

The frequency of the LSE oscillator must exceed 30 kHz to avoid false positive detections.

### 5.2.8 ADC clock

The ADC clock (refer to the device datasheet for maximum frequency) is derived from the system clock SYSCLK or from the kernel clock output HSIKER (see ADCSEL[1:0] bitfield of the [RCC peripherals independent clock configuration register \(RCC\\_CCIPR\)](#)). It can be prescaled by 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256, by configuring the PRESC[3:0] bitfield of the ADC\_CCR register. It is asynchronous to the APB clock. Alternatively, the ADC clock can be derived from the APB clock of the ADC bus interface (synchronous clock), divided by a programmable factor (1, 2 or 4) set through the CKMODE[1:0] bitfield of the ADC\_CFGR2 register.

### 5.2.9 RTC clock

The RTCCLK clock source can be either the HSE/32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the [RCC control/status register 1 \(RCC\\_CSR1\)](#). This selection cannot be modified without resetting the RTC domain. The system must always be

configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC.

RTC does not operate if the  $V_{DD}$  supply is powered off or if the internal voltage regulator is powered off (removing power from the  $V_{CORE}$  domain).

When the RTC clock is LSE or LSI, the RTC remains clocked and functional under system reset.

### 5.2.10 Timer clock

The timer clock TIMPCLK is derived from PCLK (used for APB) as follows:

1. If the APB prescaler is set to 1, TIMPCLK frequency is equal to PCLK frequency.
2. Otherwise, the TIMPCLK frequency is set to twice the PCLK frequency.

### 5.2.11 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.

### 5.2.12 Clock-out capability

#### MCO and MCO2

The MCO and MCO2 pins output, independently of each other, the clock selected from:

- LSI
- LSE
- SYSCLK
- HSI48
- HSE

The multiplexers for MCO and MCO2, respectively, are controlled by the MCSEL[2:0] and MC2SEL[2:0] bitfields of the [RCC clock configuration register \(RCC\\_CFGR\)](#). Their outputs are further divided by a factor set through the MCOPRE[2:0] and MC2PRE[2:0] bitfields of the [RCC clock configuration register \(RCC\\_CFGR\)](#).

#### LSCO

The LSCO pin allows outputting one of low-speed clocks:

- LSI
- LSE

The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN bit of the [RCC control/status register 1 \(RCC\\_CSR1\)](#). The configuration registers of the corresponding GPIO port must be programmed in alternate function mode.

This function remains available in Stop mode.

### 5.2.13 Internal/external clock measurement with TIM14/TIM16/TIM17

It is possible to indirectly measure the frequency of all on-board clock sources with the TIM14, TIM16 and TIM17 channel 1 input capture, as represented in [Figure 11](#), [Figure 12](#) and [Figure 13](#).

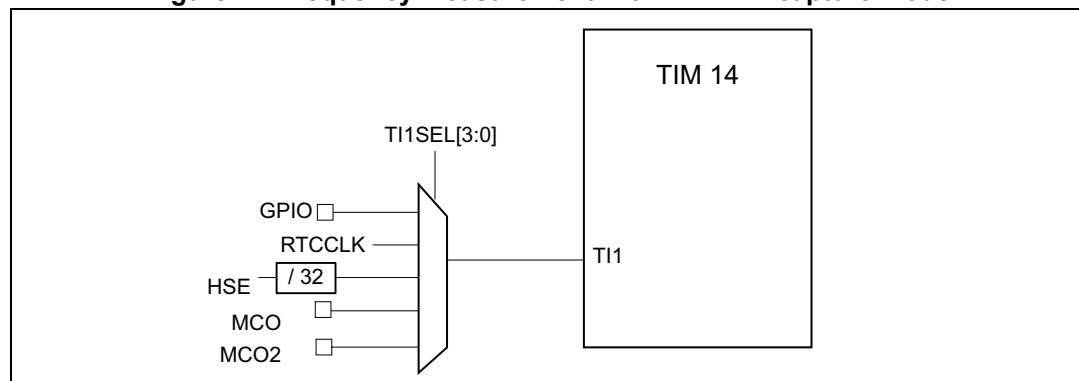
#### TIM14

By setting the TI1SEL[3:0] field of the TIM14\_TISEL register, the clock selected for the input capture channel1 of TIM14 can be one of:

- GPIO (refer to the alternate function mapping in the device datasheets)
- RTC clock (RTCCLK)
- HSE clock divided by 32
- MCO (MCU clock output)
- MCO2 (MCU clock output)

MCO and MCO2 are controlled by the MCOSEL[2:0] and MCO2SEL[2:0] bitfields, respectively, of the clock configuration register (RCC\_CFGR). All clock sources can be selected for the MCO and MCO2 pins.

**Figure 11. Frequency measurement with TIM14 in capture mode**

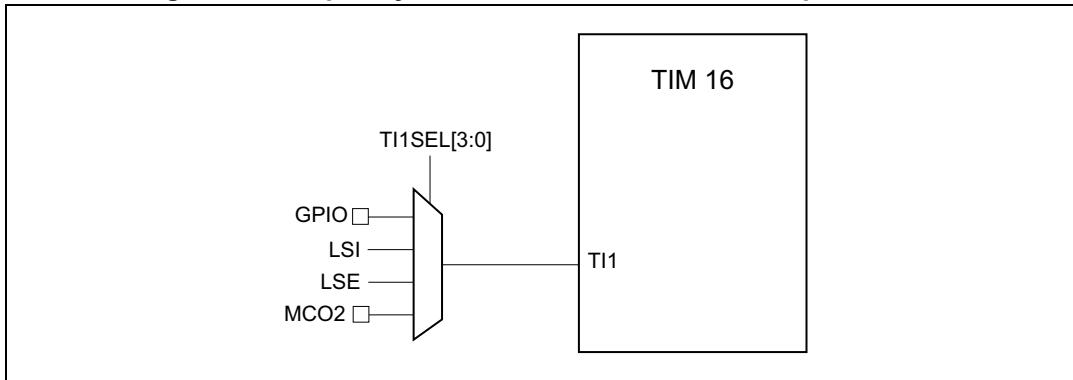


#### TIM16

By setting the TI1SEL[3:0] field of the TIM16\_TISEL register, the clock selected for the input capture channel1 of TIM16 can be one of:

- GPIO (refer to the alternate function mapping in the device datasheets).
- LSI clock
- LSE clock
- MCO2

MCO2 is controlled by the MCO2SEL[2:0] bitfield of the clock configuration register (RCC\_CFGR). All clock sources can be selected for the MCO2 pin.

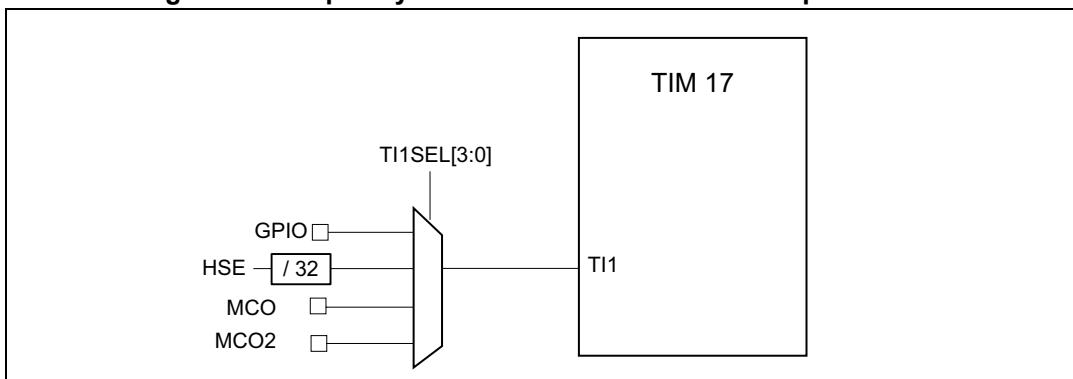
**Figure 12. Frequency measurement with TIM16 in capture mode**

### TIM17

By setting the TI1SEL[3:0] field of the TIM17\_TISEL register, the clock selected for the input capture channel1 of TIM17 can be one of:

- GPIO Refer to the alternate function mapping in the device datasheets.
- HSE clock divided by 32
- MCO (MCU clock output)
- MCO2 (MCU clock output)

MCO and MCO2 are controlled by the MCOSEL[2:0] and MCO2SEL[2:0] bitfields, respectively, of the clock configuration register (RCC\_CFGR). All clock sources can be selected for the MCO and MCO2 pins.

**Figure 13. Frequency measurement with TIM17 in capture mode**

### Calibration of the HSI48 oscillator

For TIM14, TIM16, and TIM17, the primary purpose of connecting the LSE to the channel 1 input capture is to precisely measure HSISYS (derived from HSI48) selected as system clock. Counting HSISYS clock pulses between consecutive edges of the LSE clock (the time reference) allows measuring the HSI48 oscillator frequency with nearly the same accuracy as the accuracy of the 32.768 kHz quartz crystal used with the LSE oscillator (typically a few tens of ppm). The HSI48 oscillator can then be trimmed to compensate for deviations from target frequency, due to manufacturing, process, temperature and/or voltage variation.

The HSI48 oscillator has dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (for example, the HSISYS/LSE ratio): the measurement accuracy is therefore closely related to the ratio between the two clock sources. Increasing the ratio allows improving the measurement accuracy.

Generated by the HSE oscillator, the HSE clock (divided by 32) used as time reference is the second best method for reaching a good HSI48 frequency measurement accuracy. It is recommended in absence of the LSE clock.

In order to further improve the precision of the HSI48 oscillator calibration, it is advised to employ one or a combination of the following measures to increase the frequency measurement accuracy:

- set the HSISYS divider to 1 for HSISYS frequency to be equal to HSI48 frequency
- average the results of multiple consecutive measurements
- use the input capture prescaler of the timer (one capture every up to eight periods)

### Measurement of the LSI oscillator frequency

The measurement of the LSI oscillator frequency uses the same principle as that for calibrating the HSI48 oscillator. TIM16 channel1 input capture must be used for LSI clock, and HSE selected as system clock source. The number of HSE clock pulses between consecutive edges of the LSI signal, counted by TIM16, is then representative of the LSI clock period.

#### 5.2.14 Peripheral clock enable registers

Each peripheral clock can be enabled by the corresponding enable bit of the RCC\_AHBENR or RCC\_APBENRx registers.

When the peripheral clock is not active, the peripheral registers read or write accesses are not supported.

**Caution:** The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral. After the enable bit is set, there is a 2-clock-cycle delay before the clock becomes active, which the software must take into account.

### 5.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU clock. The memory interface clocks (Flash memory and SRAM interfaces) can be stopped by software during sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop mode stops all the clocks in the V<sub>CORE</sub> domain and disable the HSI48 and HSE oscillators.

The USART1 and I2C1 peripherals can enable the HSI48 oscillator even when the MCU is in Stop mode (if HSI48 is selected as clock source for one of those peripherals).

The USART1 peripheral can also operate with the clock from the LSE oscillator when the system is in Stop mode, if LSE is selected as clock source for that peripheral and the LSE oscillator is enabled (LSEON set). In that case, the LSE oscillator remains

active when the device enters Stop mode (these peripherals do not have the capability to turn on the LSE oscillator).

- Standby and Shutdown modes stop all clocks in the V<sub>CORE</sub> domain and disable the HSI48, and HSE oscillators.

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

When leaving the Stop mode, HSISYS becomes automatically the system clock.

When leaving the Standby and Shutdown modes, HSISYS (with frequency equal to HSI48/4) becomes automatically the system clock. At wakeup from Standby and Shutdown mode, the user trim is lost.

If a Flash memory programming operation is ongoing, Stop, Standby, and Shutdown entry is delayed until the Flash memory interface access is finished. If an access to the APB domain is ongoing, the Stop, Standby, and Shutdown entry is delayed until the APB access is finished.

## 5.4 RCC registers

Unless otherwise specified, the RCC registers support word, half-word, and byte access, without any wait state.

### 5.4.1 RCC clock control register (RCC\_CR)

Address offset: 0x00

Power-on reset value: 0x0000 0500

Other types of reset: same as power-on reset, except HSEBYP bit that keeps its previous value.

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.	CSS ON	HSE BYP	HSE RDY	HSE ON
												rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HSIDIV[2:0]			HSI RDY	HSI KERON	HSION	HSIKERDIV[2:0]			Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	r	rw	rw	rw	rw	rw					

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

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

Set by software to enable the clock security system. When the bit 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. The bit is cleared by hardware upon reset.

0: Disable

1: Enable

Bit 18 **HSEBYP**: HSE crystal oscillator bypass

Set and cleared by software.

When the bit is set, the internal HSE oscillator is bypassed for use of an external clock. The external clock must then be enabled with the HSEON bit set. Write access to the bit is only effective when the HSE oscillator is disabled.

0: No bypass

1: Bypass

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable and ready for use.

0: Not ready

1: Ready

*Note: Upon clearing HSEON, HSERDY goes low after six HSE clock cycles.*

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop, or Standby, or Shutdown mode. This bit cannot be cleared if the HSE oscillator is used directly or indirectly as the system clock.

0: Disable

1: Enable

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:11 **HSIDIV[2:0]**: HSI48 clock division factor

This bitfield controlled by software sets the division factor of the HSI48 clock divider to produce HSISYS clock:

000: 1

001: 2

010: 4 (reset value)

011: 8

100: 16

101: 32

110: 64

111: 128

Bit 10 **HSIRDY**: HSI48 clock ready flag

Set by hardware when the HSI48 oscillator is enabled through HSION and ready to use (stable).

0: Not ready

1: Ready

*Note: Upon clearing HSION, HSIRDY goes low after six HSI48 clock cycles.*

Bit 9 **HSIKERON**: HSI48 always-enable for peripheral kernels.

Set and cleared by software.

Setting the bit activates the HSI48 oscillator in Run and Stop modes, regardless of the HSION bit state. The HSI48 clock can only feed USART1, USART2, and I2C1 peripherals configured with HSI48 as kernel clock.

0: HSI48 oscillator enable depends on the HSION bit

1: HSI48 oscillator is active in Run and Stop modes

*Note: Keeping the HSI48 active in Stop mode allows speeding up the serial interface communication as the HSI48 clock is ready immediately upon exiting Stop mode.*

**Bit 8 HSION:** HSI48 clock enable

Set and cleared by software and hardware, with hardware taking priority.

Kept low by hardware as long as the device is in a low-power mode.

Kept high by hardware as long as the system is clocked with a clock derived from HSI48.

This includes the exit from low-power modes and the system clock fall-back to HSI48 upon failing HSE oscillator clock selected as system clock source.

0: Disable

1: Enable

**Bits 7:5 HSIKERDIV[2:0]:** HSI48 kernel clock division factor

This bitfield controlled by software sets the division factor of the kernel clock divider to produce HSIKER clock:

000: 1

001: 2

010: 3 (reset value)

011: 4

100: 5

101: 6

110: 7

111: 8

Bits 4:0 Reserved, must be kept at reset value.

#### 5.4.2 RCC internal clock source calibration register (RCC\_ICSCR)

Address offset: 0x04

Reset value: 0x0000 40XX

The X nibbles of the reset can vary from part to part as they depend on the part calibration in the factory.

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.	HSITRIM[6:0]							HSICAL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

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

**Bits 14:8 HSITRIM[6:0]:** HSI48 clock trimming

The value of this bitfield contributes to the HSICAL[7:0] bitfield value.

It allows HSI48 clock frequency user trimming.

The HSI48 frequency accuracy as stated in the device datasheet applies when this bitfield is left at its reset value.

**Bits 7:0 HSICAL[7:0]:** HSI48 clock calibration

This bitfield directly acts on the HSI48 clock frequency. Its value is a sum of an internal factory-programmed number and the value of the HSITRIM[6:0] bitfield. In the factory, the internal number is set to calibrate the HSI48 clock frequency to 48 MHz (with HSITRIM[6:0] left at its reset value). Refer to the device datasheet for HSI48 calibration accuracy and for the frequency trimming granularity.

*Note: The trimming effect presents discontinuities at HSICAL[7:0] multiples of 64.*

### 5.4.3 RCC clock configuration register (RCC\_CFGR)

One or two wait states are inserted when accessing this register upon a clock source switch, and between zero and 15 wait states upon updating APB or AHB prescaler values.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			Res.	MCOSEL[2:0]			Res.	MCO2PRE[2:0]			Res.	MCO2SEL[2:0]		
	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.	PPRE[2:0]			HPRE[3:0]				Res.	Res.	SWS[2:0]			SW[2:0]		
	rw	rw	rw	rw	rw	rw	rw			r	r	r	rw	rw	rw

Bits 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]**: Microcontroller clock output prescaler

This bitfield is controlled by software. It sets the division factor of the clock sent to the MCO output as follows:

000: 1

001: 2

010: 4

...

111: 128

It is highly recommended to set this field before the MCO output is enabled.

Bit 27 Reserved, must be kept at reset value.

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

This bitfield is controlled by software. It sets the clock selector for MCO output as follows:

000: no clock

001: SYSCLK

010: Reserved

011: HSI48

100: HSE

101: Reserved

110: LSI

111: LSE

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

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **MCO2PRE[2:0]**: Microcontroller clock output 2 prescaler

This bitfield is controlled by software. It sets the division factor of the clock sent to the MCO2 output as follows:

000: 1

001: 2

010: 4

...

111: 128

It is highly recommended to set this field before the MCO2 output is enabled.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **MCO2SEL[2:0]**: Microcontroller clock output 2 clock selector

This bitfield is controlled by software. It sets the clock selector for MCO2 output as follows:

- 000: no clock
- 001: SYSCLK
- 010: Reserved
- 011: HSI48
- 100: HSE
- 101: Reserved
- 110: LSI
- 111: LSE

*Note: This clock output may have some truncated cycles at startup or during MCO2 clock source switching.*

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PPRE[2:0]**: APB prescaler

This bitfield is controlled by software. To produce PCLK clock, it sets the division factor of HCLK clock as follows:

- 0xx: 1
- 100: 2
- 101: 4
- 110: 8
- 111: 16

Bits 11:8 **HPRE[3:0]**: AHB prescaler

This bitfield is controlled by software. To produce HCLK clock, it sets the division factor of SYSCLK clock as follows:

- 0xx: 1
- 1000: 2
- 1001: 4
- 1010: 8
- 1011: 16
- 1100: 64
- 1101: 128
- 1110: 256
- 1111: 512

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

Bits 5:3 **SWS[2:0]**: System clock switch status

This bitfield is controlled by hardware to indicate the clock source used as system clock:

000: HSISYS

001: HSE

010: Reserved

011: LSI

100: LSE

Others: Reserved

Bits 2:0 **SW[2:0]**: System clock switch

This bitfield is controlled by software and hardware. The bitfield selects the clock for SYCLK as follows:

000: HSISYS

001: HSE

010: Reserved

011: LSI

100: LSE

Others: Reserved

The setting is forced by hardware to 000 (HSISYS selected) when the MCU exits Stop, or Standby, or Shutdown mode, or when the setting is 001 (HSE selected) and HSE oscillator failure is detected.

**5.4.4 RCC clock interrupt enable register (RCC\_CIER)**

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.	HSE RDYIE	HSI RDYIE	Res.	LSE RDYIE	LSI RDYIE										
											rw	rw		rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **HSERDYIE**: HSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization:

0: Disable

1: Enable

Bit 3 **HSIRDYIE**: HSI48 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI48 oscillator stabilization:

0: Disable

1: Enable

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYIE**: LSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization:

0: Disable

1: Enable

Bit 0 **LSIRDYIE**: LSI ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI oscillator stabilization:

0: Disable

1: Enable

#### 5.4.5 RCC clock interrupt flag register (RCC\_CIFR)

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.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSE CSSF	CSSF	Res.	Res.	Res.	HSE RDYF	HSI RDYF	Res.	LSE RDYF	LSI RDYF
						r	r				r	r		r	r

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

Bit 9 **LSECSSF**: LSE clock security system interrupt flag

This flag indicates a pending interrupt upon LSE clock failure.

Set by hardware when a failure is detected in the LSE oscillator.

Cleared by software by setting the LSECSSC bit.

0: Interrupt not pending

1: Interrupt pending

Bit 8 **CSSF**: HSE clock security system interrupt flag

This flag indicates a pending interrupt upon HSE clock failure.

Set by hardware when a failure is detected in the HSE oscillator.

Cleared by software setting the CSSC bit.

0: Interrupt not pending

1: Interrupt pending

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

Bit 4 **HSERDYF**: HSE ready interrupt flag

This flag indicates a pending interrupt upon HSE clock getting ready.

Set by hardware when the HSE clock becomes stable and HSERDYIE is set.

Cleared by software setting the HSERDYC bit.

0: Interrupt not pending

1: Interrupt pending

Bit 3 **HSIRDYF**: HSI48 ready interrupt flag

This flag indicates a pending interrupt upon HSI48 clock getting ready.

Set by hardware when the HSI48 clock becomes stable and HSIRDYIE is set in response to setting the HSION (refer to [RCC clock control register \(RCC\\_CR\)](#)). When HSION is not set but the HSI48 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: Interrupt not pending

1: Interrupt pending

## Bit 2 Reserved, must be kept at reset value.

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

This flag indicates a pending interrupt upon LSE clock getting ready.

Set by hardware when the LSE clock becomes stable and LSERDYDIE is set.

Cleared by software setting the LSERDYC bit.

0: Interrupt not pending

1: Interrupt pending

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

This flag indicates a pending interrupt upon LSE clock getting ready.

Set by hardware when the LSI clock becomes stable and LSIRDYDIE is set.

Cleared by software setting the LSIRDYC bit.

0: Interrupt not pending

1: Interrupt pending

**5.4.6 RCC clock interrupt clear register (RCC\_CICR)**

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.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSE CSSC	CSSC	Res.	Res.	Res.	HSE RDYC	HSI RDYC	Res.	LSE RDYC	LSI RDYC
						w	w				w	w		w	w

RCC

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

Bit 9 **LSECSSC**: LSE Clock security system interrupt clear

This bit is set by software to clear the LSECSSF flag.

0: No effect

1: Clear LSECSSF flag

Bit 8 **CSSC**: Clock security system interrupt clear

This bit is set by software to clear the HSECSSF flag.

0: No effect

1: Clear CSSF flag

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

Bit 4 **HSERDYC**: HSE ready interrupt clear

This bit is set by software to clear the HSERDYF flag.

0: No effect

1: Clear HSERDYF flag

Bit 3 **HSIRDYC**: HSI48 ready interrupt clear

This bit is set software to clear the HSIRDYF flag.

0: No effect

1: Clear HSIRDYF flag

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYC**: LSE ready interrupt clear

This bit is set by software to clear the LSERDYF flag.

0: No effect

1: Clear LSERDYF flag

Bit 0 **LSIRDYC**: LSI ready interrupt clear

This bit is set by software to clear the LSIRDYF flag.

0: No effect

1: Clear LSIRDYF flag

**5.4.7 RCC I/O port reset register (RCC\_IOPRSTR)**

Address offset: 0x24

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.	GPIOF RST	Res.	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST									
										rw		rw	rw	rw	rw

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

Bit 5 **GPIOFRST**: I/O port F reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port F

Bit 4 Reserved, must be kept at reset value.

Bit 3 **GPIODRST**: I/O port D reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port D

Bit 2 **GPIOCRST**: I/O port C reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port C

Bit 1 **GPIOBRST**: I/O port B reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port B

Bit 0 **GPIOARST**: I/O port A reset

This bit is set and cleared by software.

0: no effect

1: Reset I/O port A

#### 5.4.8 RCC AHB peripheral reset register (RCC\_AHBRSTR)

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
Res.	Res.	Res.	CRC RST	Res.	Res.	Res.	FLASH RST	Res.	DMA1 RST						
			rw				rw								rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC reset

Set and cleared by software.

0: No effect

1: Reset CRC

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHRST**: Flash memory interface reset

Set and cleared by software.

0: No effect

1: Reset Flash memory interface

This bit can only be set when the Flash memory is in power down mode.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **DMA1RST**: DMA1 and DMAMUX reset

Set and cleared by software.

0: No effect

1: Reset DMA1 and DMAMUX

#### 5.4.9 RCC APB peripheral reset register 1 (RCC\_APBRSTR1)

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.	PWR RST	DBG RST	Res.	Res.	Res.	Res.	Res.	I2C1 RST	Res.	Res.	Res.	USART2 RST	Res.
			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.	TIM3 RST	Res.
														rw	

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **PWRRST**: Power interface reset

Set and cleared by software.

0: No effect

1: Reset PWR

Bit 27 **DBGRST**: Debug support reset

Set and cleared by software.

0: No effect

1: Reset DBG

Bits 26:22 Reserved, must be kept at reset value.

Bit 21 **I2C1RST**: I2C1 reset

Set and cleared by software.

0: No effect

1: Reset I2C1

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2RST**: USART2 reset

Set and cleared by software.

0: No effect

1: Reset USART2

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **TIM3RST**: TIM3 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM3

Bit 0 Reserved, must be kept at reset value.

#### 5.4.10 RCC APB peripheral reset register 2 (RCC\_APBRSTR2)

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.	ADC RST	Res.	TIM17 RST	TIM16 RST	Res.
											rw		rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM14 RST	USART1 RST	Res.	SPI1 RST	TIM1 RST	Res.	Res.	Res.	Res.	SYS CFG RST						
rw	rw		rw	rw											rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADCRST**: ADC reset

Set and cleared by software.

0: No effect

1: Reset ADC

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17RST**: TIM16 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 Reserved, must be kept at reset value.

Bit 15 **TIM14RST**: TIM14 timer reset

Set and cleared by software.

0: No effect

1: Reset TIM14 timer

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

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST**: SYSCFG reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset SYSCFG

#### 5.4.11 RCC I/O port clock enable register (RCC\_IOPENR)

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.	Res.	Res.	Res.	Res.										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GPIOF EN	Res.	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN									
										rw		rw	rw	rw	rw

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

Bit 5 **GPIOFEN**: I/O port F clock enable  
 This bit is set and cleared by software.  
 0: Disable  
 1: Enable

Bit 4 Reserved, must be kept at reset value.

Bit 3 **GPIODEN**: I/O port D clock enable  
 This bit is set and cleared by software.  
 0: Disable  
 1: Enable

Bit 2 **GPIOCEN**: I/O port C clock enable  
 This bit is set and cleared by software.  
 0: Disable  
 1: Enable

Bit 1 **GPIOBEN**: I/O port B clock enable  
 This bit is set and cleared by software.  
 0: Disable  
 1: Enable

Bit 0 **GPIOAEN**: I/O port A clock enable  
 This bit is set and cleared by software.  
 0: Disable  
 1: Enable

### 5.4.12 RCC AHB peripheral clock enable register (RCC\_AHBENR)

Address offset: 0x38

Reset value: 0x0000 0100

This register individually enables clocks to AHB peripherals. In Sleep and Stop modes, a clock enabled through this register is only supplied to the peripheral if the corresponding bit of the RCC\_AHBSMENR register is also set.

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.	CRC EN	Res.	Res.	Res.	FLASH EN	Res.	DMA1 EN						
			rw				rw								rw

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

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHEN**: Flash memory interface clock enable

Set and cleared by software.

0: Disable

1: Enable

This bit can only be cleared when the Flash memory is in power down mode.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **DMA1EN**: DMA1 and DMAMUX clock enable

Set and cleared by software.

0: Disable

1: Enable

DMAMUX is enabled as long as at least one DMA peripheral is enabled.

### 5.4.13 RCC APB peripheral clock enable register 1 (RCC\_APBENR1)

Address offset: 0x3C

Reset value: 0x0000 0000

This register individually enables clocks to APB peripherals. In Sleep and Stop modes, a clock enabled through this register is only supplied to the peripheral if the corresponding bit of the RCC\_APBSMENR1 register is also set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	PWR EN	DBG EN	Res.	Res.	Res.	Res.	Res.	I2C1 EN	Res.	Res.	Res.	USART2 EN	Res.
			rw	rw						rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WWDG EN	RTC APB EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3 EN	Res.
				rw	rw										rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **PWREN**: Power interface clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 27 **DBGEN**: Debug support clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 26:22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: I2C1 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2EN**: USART2 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG clock enable

Set by software to enable the window watchdog clock. Cleared by hardware system reset

0: Disable

1: Enable

This bit can also be set by hardware if the WWDG\_SW option bit is 0.

Bit 10 **RTCAPBEN**: RTC APB clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 9:2 Reserved, must be kept at reset value.

Bit 1 **TIM3EN**: TIM3 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 0 Reserved, must be kept at reset value.

### 5.4.14 RCC APB peripheral clock enable register 2(RCC\_APBENR2)

Address offset: 0x40

Reset value: 0x0000 0000

This register individually enables clocks to APB peripherals. In Sleep and Stop modes, a clock enabled through this register is only supplied to the peripheral if the corresponding bit of the RCC\_APBSMENR2 register is also set.

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.	ADC EN	Res.	TIM17 EN	TIM16 EN	Res.
											rw		rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM14 EN	USART1 EN	Res.	SPI1 EN	TIM1 EN	Res.	Res.	Res.	Res.	SYS CFG EN						
rw	rw		rw	rw											rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADCEN**: ADC clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17EN**: TIM16 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 17 **TIM16EN**: TIM16 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 16 Reserved, must be kept at reset value.

Bit 15 **TIM14EN**: TIM14 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 14 **USART1EN**: USART1 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 clock enable

Set and cleared by software.

0: Disable

1: Enable

Bit 11 **TIM1EN**: TIM1 timer clock enable

Set and cleared by software.

0: Disable

1: Enable

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGGEN**: SYSCFG clock enable

Set and cleared by software.

0: Disable

1: Enable

#### 5.4.15 RCC I/O port in Sleep mode clock enable register (RCC\_IOPSMENR)

Address offset: 0x44

Reset value: 0x0000 002F

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.	GPIOF SMEN	Res.	GPIOD SMEN	GPIOC SMEN	GPIOB SMEN	GPIOA SMEN									
										rw		rw	rw	rw	rw

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

Bit 5 **GPIOFSMEN**: I/O port F clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 4 Reserved, must be kept at reset value.

Bit 3 **GPIODSMEN**: I/O port D clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 2 **GPIOCSMEN**: I/O port C clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 1 **GPIOBSMEN**: I/O port B clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 0 **GPIOASMEN**: I/O port A clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

### 5.4.16 RCC AHB peripheral clock enable in Sleep/Stop mode register (RCC\_AHBSMENR)

Address offset: 0x48

Reset value: 0x0000 1301

This register can individually program which AHB peripheral clocks are disabled (bit cleared) upon the device entering Sleep or Stop mode. When a bit of this register is set (enable), the corresponding peripheral clock is supplied in Sleep or Stop mode according to the setting of the RCC\_AHBENR register.

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.	CRC SMEN	Res.	Res.	SRAM SMEN	FLASH SMEN	Res.	DMA1 SMEN						
			rw			rw	rw								rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAMSMEN**: SRAM clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 8 **FLASHSMEN**: Flash memory interface clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

This bit can be activated only when the Flash memory is in power down mode.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **DMA1SMEN**: DMA1 and DMAMUX clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Clock to DMAMUX during Sleep mode is enabled as long as the clock in Sleep mode is enabled to at least one DMA peripheral.

### 5.4.17 RCC APB peripheral clock enable in Sleep/Stop mode register 1 (RCC\_APBSMENR1)

Address offset: 0x4C

Reset value: 0x18220C02

This register can individually program which APB peripheral clocks are disabled (bit cleared) upon the device entering Sleep or Stop mode. When a bit of this register is set (enable), the corresponding peripheral clock is supplied in Sleep or Stop mode according to the setting of the RCC\_APBENR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	PWR SMEN	DBG SMEN	Res.	Res.	Res.	Res.	Res.	I2C1 SMEN	Res.	Res.	Res.	USART2 SMEN	Res.
			rw	rw						rw				rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WWDG SMEN	RTC APB SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3 SMEN	Res.
				rw	rw									rw	

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **PWRSMEN**: Power interface clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 27 **DBGSMEN**: Debug support clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 26:22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN**: I2C1 clock enable during Sleep and Stop modes

Set and cleared by software.

0: Disable

1: Enable

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2SMEN**: USART2 clock enable during Sleep and Stop modes

Set and cleared by software.

0: Disable

1: Enable

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSMEN**: WWDG clock enable during Sleep and Stop modes

Set and cleared by software.

0: Disable

1: Enable

Bit 10 **RTCAPBSMEN**: RTC APB clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 9:2 Reserved, must be kept at reset value.

Bit 1 **TIM3SMEN**: TIM3 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 0 Reserved, must be kept at reset value.

#### 5.4.18 RCC APB peripheral clock enable in Sleep/Stop mode register 2 (RCC\_APBSMENR2)

Address offset: 0x50

Reset value: 0x0017 D801

This register can individually program which APB peripheral clocks are disabled (bit cleared) upon the device entering Sleep or Stop mode. When a bit of this register is set (enable), the corresponding peripheral clock is supplied in Sleep or Stop mode according to the setting of the RCC\_APBENR2 register.

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.	ADC SMEN	Res.	TIM17 SMEN	TIM16 SMEN	Res.
											rw		rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM14 SMEN	USART1 SMEN	Res.	SPI1 SMEN	TIM1 SMEN	Res.	Res.	Res.	Res.	SYS CFG SMEN						
rw	rw		rw	rw											rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ADCSMEN**: ADC clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 19 Reserved, must be kept at reset value.

Bit 18 **TIM17SMEN**: TIM16 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 17 **TIM16SMEN**: TIM16 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 16 Reserved, must be kept at reset value.

Bit 15 **TIM14SMEN**: TIM14 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 14 **USART1SMEN**: USART1 clock enable during Sleep and Stop modes

Set and cleared by software.

0: Disable

1: Enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN**: SPI1 clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 11 **TIM1SMEN**: TIM1 timer clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGSMEN**: SYSCFG clock enable during Sleep and Stop modes

Set and cleared by software.

0: Disable

1: Enable

#### 5.4.19 RCC peripherals independent clock configuration register (RCC\_CCIPR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCSEL[1:0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2S1SEL[1:0]	I2C1SEL[1:0]	Res.	USART1SEL [1:0]												
rw	rw	rw	rw											rw	rw

Bits 31:30 **ADCSEL[1:0]**: ADCs clock source selection

This bitfield is controlled by software to select the asynchronous clock source for ADC:

00: System clock

01: Reserved

10: HSIKER

11: Reserved

Bits 29:16 Reserved, must be kept at reset value.

Bits 15:14 **I2S1SEL[1:0]**: I2S1 clock source selection

This bitfield is controlled by software to select I2S1 clock source as follows:

00: SYSCLK

01: Reserved

10: HSIKER

11: I2S\_CKIN

Bits 13:12 **I2C1SEL[1:0]**: I2C1 clock source selection

This bitfield is controlled by software to select I2C1 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSIKER
- 11: Reserved

Bits 11:2 Reserved, must be kept at reset value.

Bits 1:0 **USART1SEL[1:0]**: USART1 clock source selection

This bitfield is controlled by software to select USART1 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSIKER
- 11: LSE

#### 5.4.20 RCC control/status register 1 (RCC\_CSR1)

Up to three wait states are inserted in case of successive accesses to this register.

The register bits are only reset upon RTC domain reset (see [Section 5.1.3: RTC domain reset](#)), except the LSCOSEL, LSCOEN, and RTCRST bits that are only reset upon RTC domain power-on reset. Any internal or external reset has no effect on these bits.

Address offset: 0x5C

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.	LSCO SEL	LSCO EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCRST
						rw	rw								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.	LSE CSSD	LSE CSSON	Res.	LSE DRV	LSE BYP	LSE RDY	LSEON	
rw						rw	rw	r	rw		rw	rw	r	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **LSCOSEL**: Low-speed clock output selection

Set and cleared by software to select the low-speed output clock:

- 0: LSI
- 1: LSE

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable

Set and cleared by software.

- 0: Disable
- 1: Enable

Bits 23:17 Reserved, must be kept at reset value.

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

Set and cleared by software to reset the RTC domain:

- 0: No effect
- 1: Reset

**Bit 15 RTCEN:** RTC clock enable

Set and cleared by software. The bit enables clock to RTC and TAMP.

0: Disable

1: Enable

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

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

Set by software to select the clock source for the RTC as follows:

00: No clock

01: LSE

10: LSI

11: HSE divided by 32

Once the RTC clock source is selected, it cannot be changed anymore unless the RTC domain is reset, or unless a failure is detected on LSE (LSECSSD is set). The RTCRST bit can be used to reset this bitfield to 00.

Bit 7 Reserved, must be kept at reset value.

**Bit 6 LSECSSD:** CSS on LSE failure Detection

Set by hardware to indicate when a failure is detected by the clock security system on the external 32 kHz oscillator (LSE):

0: No failure detected

1: Failure detected

**Bit 5 LSECSSON:** CSS on LSE enable

Set by software to enable the clock security system on LSE (32 kHz) oscillator as follows:

0: Disable

1: Enable

LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected.

Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD =1). In that case the software **must** disable the LSECSSON bit.

Bit 4 Reserved, must be kept at reset value.

**Bits 3 LSEDRV:** LSE oscillator drive capability

Set by software to select the LSE oscillator drive capability as follows:

0: medium-high driving capability

1: high driving capability

Applicable when the LSE oscillator is in Xtal mode, as opposed to bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass the LSE oscillator (in debug mode).

0: Not bypassed

1: Bypassed

This bit can be written only when the external 32 kHz oscillator is disabled (LSEON=0 and LSERDY=0).

Bit 1 **LSERDY**: LSE oscillator ready

Set and cleared by hardware to indicate when the external 32 kHz oscillator is ready (stable):

0: Not ready

1: Ready

After the LSEON bit is cleared, LSERDY goes low after 6 external low-speed oscillator clock cycles.

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software to enable LSE oscillator:

0: Disable

1: Enable

**5.4.21 RCC control/status register 2 (RCC\_CSR2)**

Up to three wait states are inserted in case of successive accesses to this register. The register is reset upon system reset, except for reset flags that are only reset upon power reset.

Address offset: 0x60

Reset value: 0xXX00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	PWR RSTF	PIN RSTF	OBL RSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r		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 reset occurs due to illegal Stop, or Standby, or Shutdown mode entry.

Cleared by setting the RMVF bit.

0: No illegal mode reset occurred

1: Illegal mode reset occurred

This operates only if nRST\_STOP, or nRST\_STDBY or nRST\_SHDW option bits are cleared.

Bit 30 **WWDGRSTF**: Window watchdog reset flag

Set by hardware when a window watchdog reset occurs.

Cleared by setting the RMVF bit.

0: No window watchdog reset occurred

1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent window watchdog reset flag

Set by hardware when an independent watchdog reset domain occurs.

Cleared by setting the RMVF bit.

0: No independent watchdog reset occurred

1: Independent watchdog reset occurred

Bit 28 **SFTRSTF**: Software reset flag

Set by hardware when a software reset occurs.

Cleared by setting the RMVF bit.

0: No software reset occurred

1: Software reset occurred

Bit 27 **PWRRSTF**: BOR or POR/PDR flag

Set by hardware when a BOR or POR/PDR occurs.

Cleared by setting the RMVF bit.

0: No BOR or POR occurred

1: BOR or POR occurred

Bit 26 **PINRSTF**: Pin reset flag

Set by hardware when a reset from the NRST pin occurs.

Cleared by setting the RMVF bit.

0: No reset from NRST pin occurred

1: Reset from NRST pin occurred

Bit 25 **OBLRSTF**: Option byte loader reset flag

Set by hardware when a reset from the Option byte loading occurs.

Cleared by setting the RMVF bit.

0: No reset from Option byte loading occurred

1: Reset from Option byte loading occurred

## Bit 24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: Remove reset flags

Set by software to clear the reset flags.

0: No effect

1: Clear reset flags

Bits 22:2 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 ready (stable):

0: Not ready

1: Ready

After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles. This bit can be set even if LSION = 0 if the LSI is requested by the Clock Security System on LSE, by the Independent Watchdog or by the RTC.

Bit 0 **LSION**: LSI oscillator enable

Set and cleared by software to enable/disable the LSI oscillator:

0: Disable

## 1: Enable

## 5.4.22 RCC register map

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

**Table 23. RCC register map and reset values**

Table 23. RCC register map and reset values (continued)

Off-set	Register						
0x20	RCC_CICR					Res.	31
0x24	RCC_IOPRSTR					Res.	30
0x28	RCC_AHBRSTR					Res.	29
0x2C	RCC_APBRSTR1					Res.	28
0x30	RCC_APBRSTR2					Res.	27
0x34	RCC_IOPENR					Res.	26
0x38	RCC_AHBENR					Res.	25
0x3C	RCC_APBENR1					Res.	24
0x40	RCC_APBENR2					Res.	23
0x44	RCC_IOPSMENR					Res.	22
	Reset value					Res.	21
0x20	RCC_CICR	Res.	Res.	Res.	Res.	Res.	20
0x24	RCC_IOPRSTR	Res.	Res.	Res.	Res.	Res.	19
0x28	RCC_AHBRSTR	Res.	Res.	Res.	Res.	Res.	18
0x2C	RCC_APBRSTR1	Res.	Res.	Res.	Res.	Res.	17
0x30	RCC_APBRSTR2	Res.	Res.	Res.	Res.	Res.	16
0x34	RCC_IOPENR	Res.	Res.	Res.	Res.	Res.	15
0x38	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	14
0x3C	RCC_APBENR1	Res.	Res.	Res.	Res.	Res.	13
0x40	RCC_APBENR2	Res.	Res.	Res.	Res.	Res.	12
0x44	RCC_IOPSMENR	Res.	Res.	Res.	Res.	Res.	11
	Reset value	Res.	Res.	Res.	Res.	Res.	10
0x20	RCC_CICR	0	0	0	0	LSECSSC	9
0x24	RCC_IOPRSTR	0	0	0	0	CSSC	8
0x28	RCC_AHBRSTR	0	0	0	0	Res.	7
0x2C	RCC_APBRSTR1	0	0	0	0	Res.	6
0x30	RCC_APBRSTR2	0	0	0	0	Res.	5
0x34	RCC_IOPENR	0	0	0	0	Res.	4
0x38	RCC_AHBENR	0	0	0	0	HSERDYC	3
0x3C	RCC_APBENR1	0	0	0	0	HSIORDYC	2
0x40	RCC_APBENR2	0	0	0	0	GPIOCRST	1
0x44	RCC_IOPSMENR	0	0	0	0	GPIOBRST	0
	Reset value	0	0	0	0	GPIOARST	0

**Table 23. RCC register map and reset values (continued)**

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

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

### 6.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIO<sub>x</sub>\_MODER, GPIO<sub>x</sub>\_OTYPER, GPIO<sub>x</sub>\_OSPEEDR and GPIO<sub>x</sub>\_PUPDR), two 32-bit data registers (GPIO<sub>x</sub>\_IDR and GPIO<sub>x</sub>\_ODR) and a 32-bit set/reset register (GPIO<sub>x</sub>\_BSRR). In addition all GPIOs have a 32-bit locking register (GPIO<sub>x</sub>\_LCKR) and two 32-bit alternate function selection registers (GPIO<sub>x</sub>\_AFRH and GPIO<sub>x</sub>\_AFRL).

### 6.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIO<sub>x</sub>\_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, I/O analog mode
- Input data to input data register (GPIO<sub>x</sub>\_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIO<sub>x</sub>\_BSRR) for bitwise write access to GPIO<sub>x</sub>\_ODR
- Locking mechanism (GPIO<sub>x</sub>\_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers (at most 16 AFs possible per I/O)
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

### 6.3 GPIO functional description

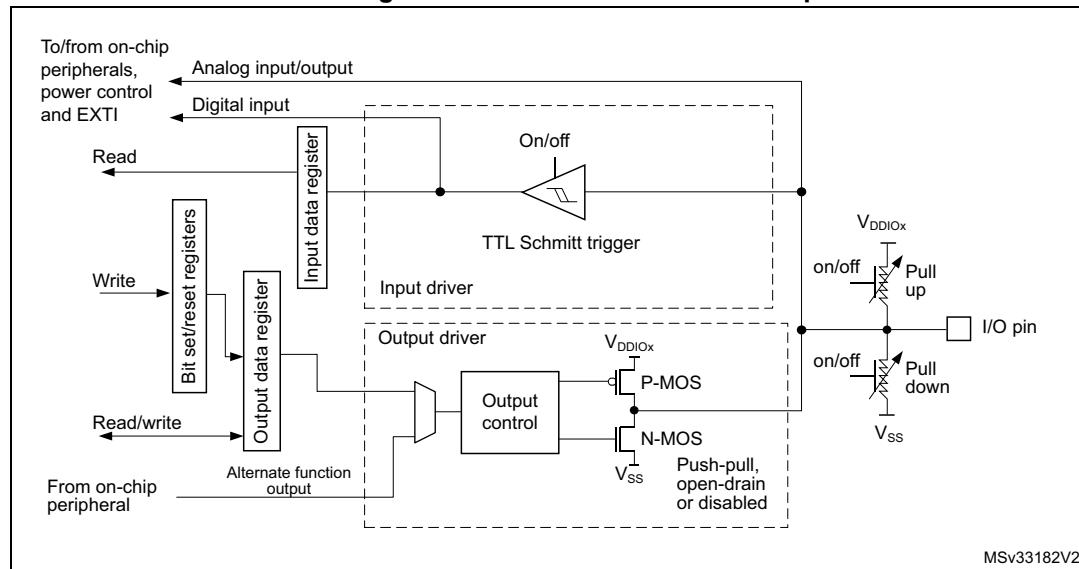
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

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

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIO<sub>x</sub>\_BSRR and GPIO<sub>x</sub>\_BRR registers is to allow atomic read/modify accesses to any of the GPIO<sub>x</sub>\_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

*Figure 14* shows the basic structures of a standard I/O port bit. *Table 24* gives the possible port bit configurations.

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



**Table 24. Port bit configuration table<sup>(1)</sup>**

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]	PUPD(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	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	1	Reserved	

**Table 24. Port bit configuration table<sup>(1)</sup> (continued)**

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
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	Reserved	

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

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

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

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

- PA14: SWCLK in pull-down
- PA13: SWDIO in pull-up

*Note:* PA14 is shared with BOOT0 functionality. Caution is required as the debugging device can manipulate BOOT0 pin value.

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.

The GPIO pins can operate as:

- **GPIO:** output, input, or analog I/O, depending on the GPIOx\_MODER register setting
- **Alternate function**

The GPIOs with debug alternate functions are set to Alternate function mode upon reset.

- **Additional function**

Available for some GPIO pins, the Additional function mode is set through the control registers of the corresponding functional block such as ADC, DAC, RTC, RCC, and PWR, regardless of the GPIOx\_MODER register setting.

When an I/O is set in Additional function mode, it is recommended to set its corresponding GPIO multiplexer in the GPIOx\_MODER register to Analog mode.

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

Each functional block signal to connect on the device pins as alternate function is internally routed towards multiple GPIO pins. Each GPIO pin has a multiplexer with 16 positions (AF0 to AF15) controlled through the GPIOx\_AFRL and GPIOx\_AFRH registers, to select one of up to 16 alternate functions at a time. The alternate function selected for a GPIO pin is physically connected to the pin through GPIO mode multiplexer controlled by the GPIOx\_MODER register.

Upon reset, the alternate function multiplexer on each GPIO is set to AF0 position.

This flexibility eases PCB routing and allows configuring small pin-count devices to match the application requirements.

The mapping of alternate function signals to GPIO alternate function multiplexers is detailed in the device datasheet.

### 6.3.3 I/O port control registers

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

### 6.3.4 I/O port data registers

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

See [Section 6.4.5: GPIO port input data register \(GPIOx\\_IDR\) \(x = A, B, C, D, F\)](#) and [Section 6.4.6: GPIO port output data register \(GPIOx\\_ODR\) \(x = A, B, C, D, F\)](#) for the register descriptions.

### 6.3.5 I/O data bitwise handling

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

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

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

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

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

### 6.3.6 GPIO locking mechanism

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

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

The LOCK sequence (refer to [Section 6.4.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A, B, C, D, F\)](#)) can only be performed using a word (32-bit long) access to the GPIOx\_LCKR register due to the fact that GPIOx\_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in [Section 6.4.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A, B, C, D, F\)](#).

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

When an I/O pin operates in Alternate function mode, the alternate function selected determines whether it acts as an input or as an output.

The pull-up/pull-down and output speed settings (via the GPIOx\_OTYPER, GPIOx\_PUPDR and GPIOx\_OSPEEDER registers, respectively) remain effective.

### 6.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the given pin must not be configured in analog mode or being used as oscillator pin, so the input trigger is kept enabled.

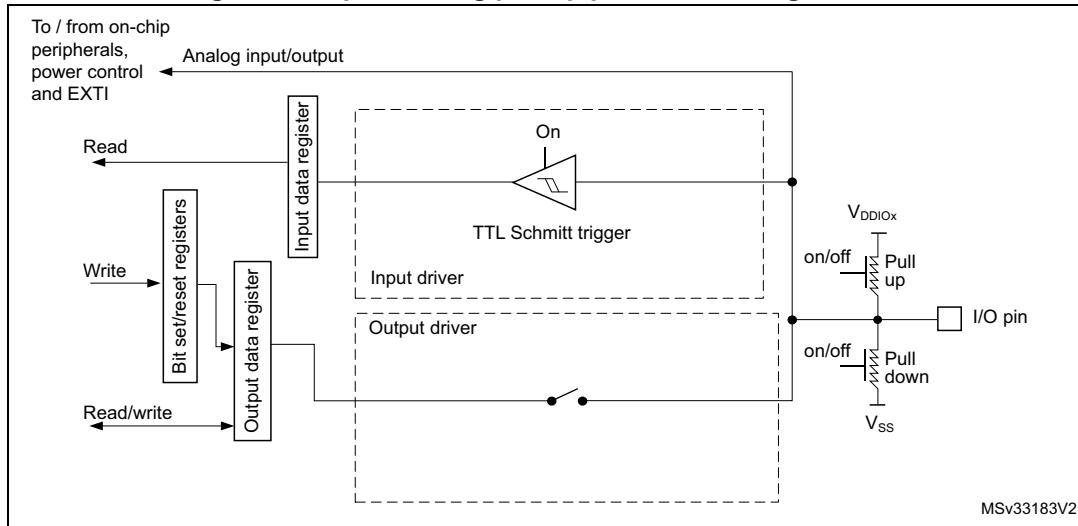
Refer to [Section 12: Extended interrupt and event controller \(EXTI\)](#).

### 6.3.9 Input configuration

When the I/O port is programmed as input:

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

[Figure 15](#) shows the input configuration of the I/O port bit.

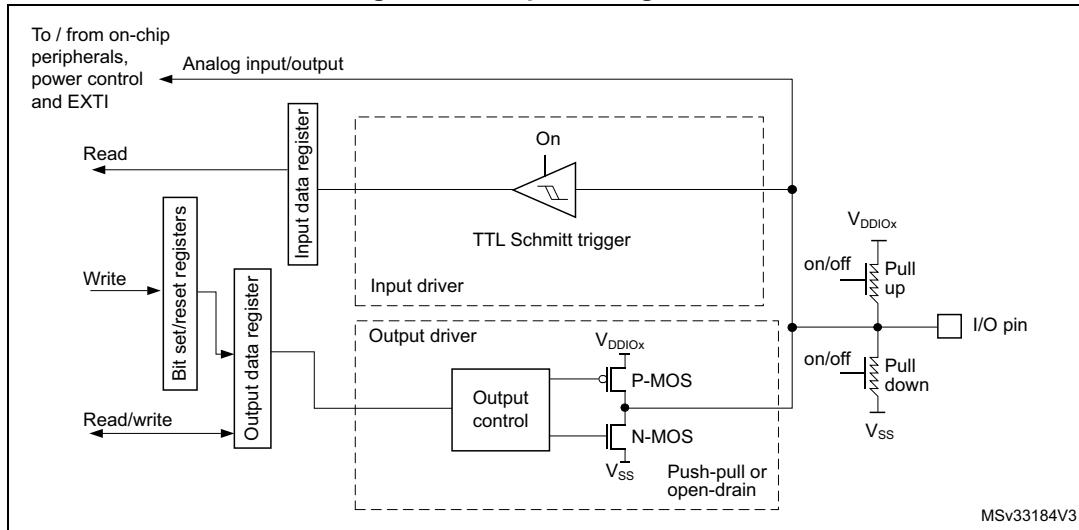
**Figure 15. Input floating/pull up/pull down configurations**

### 6.3.10 Output configuration

When the I/O port is programmed as output:

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

*Figure 16* shows the output configuration of the I/O port bit.

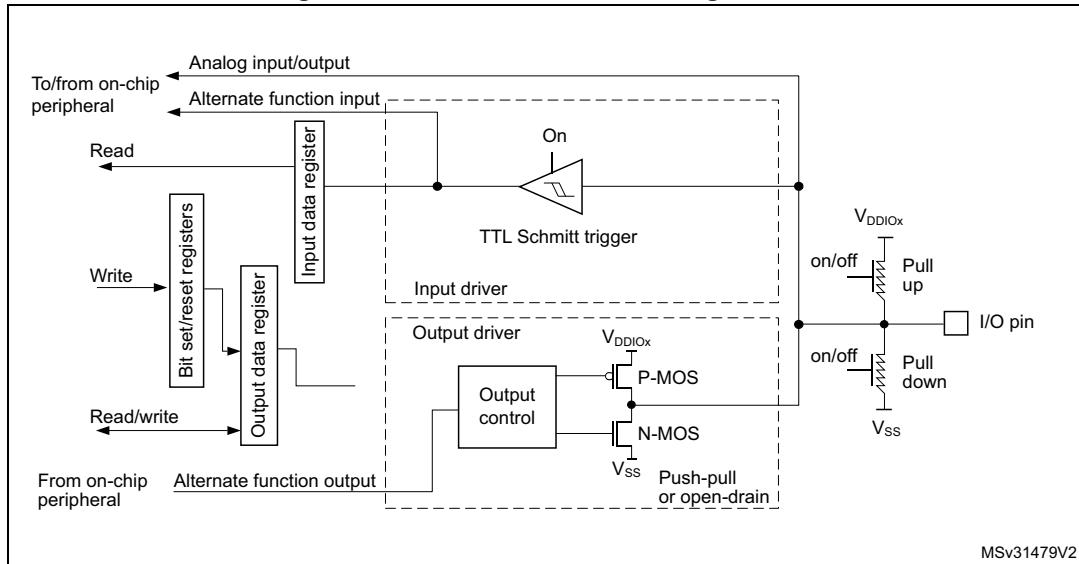
**Figure 16. Output configuration**

### 6.3.11 Alternate function configuration

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

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

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

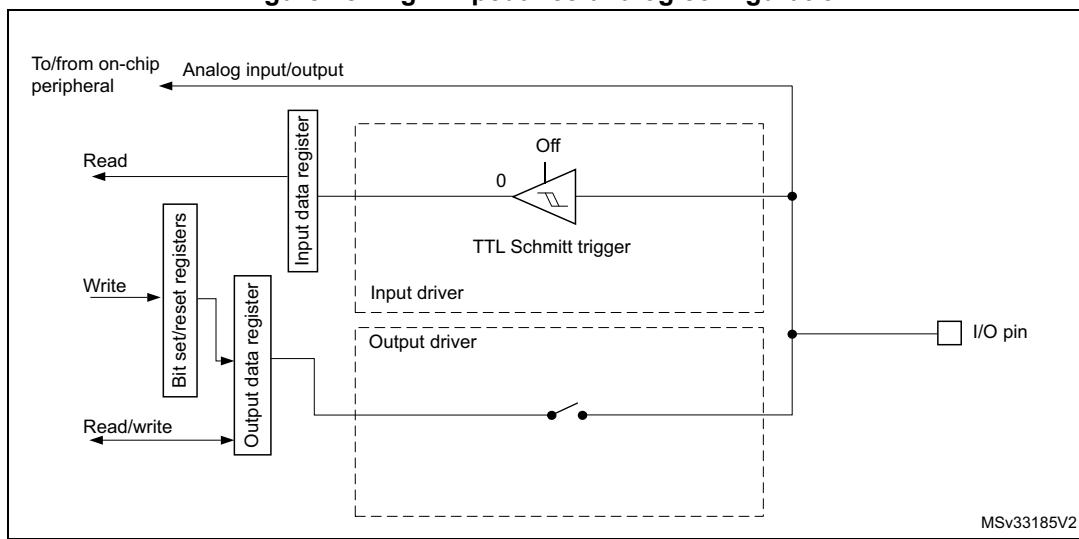
**Figure 17. Alternate function configuration-**

### 6.3.12 Analog configuration

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

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

*Figure 18* shows the high-impedance, analog configuration of the I/O port bits.

**Figure 18. High impedance-analog configuration**

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

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

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

When HSE or LSE oscillator is bypassed, its input pin is used as external clock input and its output pin is free for use as GPIO.

For the devices housed in 48-pin packages, the HSE and LSE oscillators have separate input and output pins (see HSE\_NOT\_REMAPPED bit of the FLASH option bytes). On packages with less than 48 pins, HSE and LSE oscillators have one common input pin OSCX\_IN and one common output pin OSCX\_OUT, which restricts their use to one at a time (the other must be disabled).

### 6.3.14 Small package adjustment

Due to the limited number of pins on SO8 package, multiple GPIOs are connected to I/O pins. The SYSCFG\_CFGR3 register allows selecting which of them is active, to prevent conflicts.

### 6.3.15 Reset pin (PF2) in GPIO mode

The PF2 pin can be configured as reset I/O or as a GPIO.

To configure PF2 as a GPIO (input, output, AF, or analog I/O), set the NOT\_RESET\_INPUT\_ONLY bit and clear the NOT\_GPIO\_MODE\_ONLY bit of the FLASH option bytes. The new setting only takes effect upon the option byte loading (OBL) event following a reset. Until the reset release, PF2 keeps acting as reset I/O.

When PF2 acts as a GPIO, reset can only be triggered from one of the device internal reset sources and the reset signal cannot be output.

For further information on reset function, refer to the *RCC* section.

### 6.3.16 Boot0 pin (PA14) in GPIO mode

The PA14 pin can be configured as Boot0 input or as a GPIO.

To configure PA14 as a GPIO, set the USE\_BOOT0\_OPT bit of the FLASH option bytes. Following this, PA14 is configured by default as serial wire debug (SWD) interface clock input (SWCLK).

For further information on boot configuration, refer to *Boot configuration* section.

## 6.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to [Table 25](#).

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

Port D is only available on STM32C03xx products.

### 6.4.1 GPIO port mode register (GPIO<sub>x</sub>\_MODER) (x = A, B, C, D, F)

Address offset: 0x00

Reset value: 0xEBFF FFFF (port A)

Reset value: 0xFFFF FFFF (ports other than A)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw										

Bits 31:0 **MODEy[1:0]**: Port x configuration for I/O y (y = 15 to 0)

These bits are written by software to set the I/O to one of four operating modes.

00: Input

01: Output

10: Alternate function

11: Analog

### 6.4.2 GPIO port output type register (GPIO<sub>x</sub>\_OTYPER) (x = A, B, C, D, 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	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
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
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 **OTy**: Port x configuration for I/O y (y = 15 to 0)

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

0: Output push-pull (reset state)

1: Output open-drain

### 6.4.3 GPIO port output speed register (GPIOx\_OSPEEDR) (x = A, B, C, D, F)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 0000 (ports other than A)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
rw	rw	rw	rw	rw	rw										

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration for I/O y (y = 15 to 0)

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

00: Very low speed

01: Low speed

10: High speed

11: Very high speed

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

The FT\_c GPIOs cannot be set to high speed.

### 6.4.4 GPIO port pull-up/pull-down register (GPIOx\_PUPDR) (x = A, B, C, D, F)

Address offset: 0x0C

Reset value: 0x2400 0000 (for port A)

Reset value: 0x0000 0000 (ports other than A)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw										

Bits 31:0 **PUPDy[1:0]**: Port x configuration I/O y (y = 15 to 0)

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

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

#### 6.4.5 GPIO port input data register (GPIOx\_IDR) (x = A, B, C, D, F)

Address offset: 0x10

Reset value: 0x0000 XXXX

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
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
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 **IDy**: Port x input data I/O y (y = 15 to 0)

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

#### 6.4.6 GPIO port output data register (GPIOx\_ODR) (x = A, B, C, D, 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.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw															

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

Bits 15:0 **ODy**: Port output data I/O y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx\_BSRR register (x = A, B, C, D, F).

#### 6.4.7 GPIO port bit set/reset register (GPIOx\_BSRR) (x = A, B, C, D, 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 I/O y (y = 15 to 0)

These bits are write-only. A read operation always returns 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 I/O y (y = 15 to 0)

These bits are write-only. A read operation always returns 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

#### 6.4.8 GPIO port configuration lock register (GPIOx\_LCKR) (x = A, B, C, D, F)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

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

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

Address offset: 0x1C

Reset value: 0x0000 0000

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
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

#### Bit 16 **LCKK:** Lock key

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

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx\_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

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

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

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

RD LCKR

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

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

*Any error in the lock sequence aborts the lock.*

*After the first lock sequence on any bit of the port, any read access on the LCKK bit returns '1' until the next MCU reset or peripheral reset.*

Bits 15:0 **LCK[15:0]:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is '0'.

0: Port configuration not locked

1: Port configuration locked

### 6.4.9 GPIO alternate function low register (GPIOx\_AFRL) (x = A, B, C, D, F)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw												

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x pin y (y = 0 to 7)

These bits are written by software to configure alternate function I/Os

0000: AF0	1000: AF8
0001: AF1	1001: AF9
0010: AF2	1010: AF10
0011: AF3	1011: AF11
0100: AF4	1100: AF12
0101: AF5	1101: AF13
0110: AF6	1110: AF14
0111: AF7	1111: AF15

### 6.4.10 GPIO alternate function high register (GPIOx\_AFRH) (x = A, B, C, D, F)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw												

Bits 31:0 **AFSELy[3:0]**: Alternate function selection for port x, I/O y (y = 8 to 15)

These bits are written by software to configure alternate function I/Os

0000: AF0	1000: AF8
0001: AF1	1001: AF9
0010: AF2	1010: AF10
0011: AF3	1011: AF11
0100: AF4	1100: AF12
0101: AF5	1101: AF13
0110: AF6	1110: AF14
0111: AF7	1111: AF15

### 6.4.11 GPIO port bit reset register (GPIOx\_BRR) (x = A, B, C, D, 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, must be kept at reset value.

Bits 15:0 **BRy**: Port x reset I/O y (y = 15 to 0)

These bits are write-only. A read operation always returns 0x0000.

0: No action on the corresponding ODx bit

1: Reset the corresponding ODx bit

### **6.4.12 GPIO register map**

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

**Table 25. GPIO register map and reset values**

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 7 System configuration controller (SYSCFG)

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

- Enabling/disabling I<sup>2</sup>C Fast Mode Plus on some I/O ports
- Configuring the IR modulation signal and its output polarity
- Remapping of some I/O ports
- Remapping the memory located at the beginning of the code area
- Flag pending interrupts from each interrupt line
- Managing robustness feature

### 7.1 SYSCFG registers

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

This register is used for specific configurations of memory and DMA requests remap and to control special I/O features.

Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the hardware BOOT selection. After reset these bits take the value selected by the actual boot mode configuration.

When FM+ mode is activated, the speed configuration of the I/O (GPIOx OSPEEDR) is ignored. Be sure to set the corresponding x\_FMP bits only after the AFO selection is set to I2C through the GPIOx\_AFRH or GPIOx\_AFRL registers.

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the actual boot mode configuration)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	I2C <sub>-</sub> PC14 <sub>-</sub> FMP <sup>(1)</sup>	I2C <sub>-</sub> PA10 <sub>-</sub> FMP	I2C <sub>-</sub> PA9 <sub>-</sub> FMP	Res.	I2C1 <sub>-</sub> FMP	I2C <sub>-</sub> PB9 <sub>-</sub> FMP <sup>(1)</sup>	I2C <sub>-</sub> PB8 <sub>-</sub> FMP <sup>(1)</sup>	I2C <sub>-</sub> PB7 <sub>-</sub> FMP	I2C <sub>-</sub> PB6 <sub>-</sub> FMP						
							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.	IR_MOD [1:0]		IR_POL	PA12_RMP	PA11_RMP	Res.	MEM_MODE [1:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

1. Only significant on devices integrating the corresponding peripheral or function, otherwise reserved.

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

Bit 24 **I2C\_PC14\_FMP:** Fast Mode Plus (FM+) enable for PC14

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on PC14 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on this I/O port can be enabled through one of I2Cx\_FMP bits. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

Bit 23 **I2C\_PA10\_FMP:** Fast Mode Plus (FM+) enable for PA10

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on PA10 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on this I/O port can be enabled through one of I2Cx\_FMP bits. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

Bit 22 **I2C\_PA9\_FMP:** Fast Mode Plus (FM+) enable for PA9

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on PA9 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on this I/O port can be enabled through one of I2Cx\_FMP bits. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **I2C1\_FMP:** Fast Mode Plus (FM+) enable for I2C1

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on I/O ports configured as I2C1 through GPIOx\_AFR registers.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on I/O ports configured as I2C1 can be enabled through their corresponding I2Cx\_FMP bit. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

Bit 19 **I2C\_PB9\_FMP:** Fast Mode Plus (FM+) enable for PB9

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on PB9 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on this I/O port can be enabled through one of I2Cx\_FMP bits. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

Bit 18 **I2C\_PB8\_FMP:** Fast Mode Plus (FM+) enable for PB8

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on PB8 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on this I/O port can be enabled through one of I2Cx\_FMP bits. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

Bit 17 **I2C\_PB7\_FMP:** Fast Mode Plus (FM+) enable for PB7

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on PB7 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on this I/O port can be enabled through one of I2Cx\_FMP bits. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

Bit 16 **I2C\_PB6\_FMP**: Fast Mode Plus (FM+) enable for PB6

This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on PB6 I/O port.

0: Disable

1: Enable

With this bit in disable state, the I<sup>2</sup>C FM+ driving capability on this I/O port can be enabled through one of I2Cx\_FMP bits. When I<sup>2</sup>C FM+ is enabled, the speed control is ignored.

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

Bits 7:6 **IR\_MOD[1:0]**: IR Modulation Envelope signal selection

This bitfield selects the signal for IR modulation envelope:

00: TIM16

01: USART1

10: USART2

11: Reserved

Bit 5 **IR\_POL**: IR output polarity selection

0: Output of IRTIM (IR\_OUT) is not inverted

1: Output of IRTIM (IR\_OUT) is inverted

Bit 4 **PA12\_RMP**: PA12 pin remapping

This bit is set and cleared by software. When set, it remaps the PA12 pin to operate as PA10 GPIO port, instead as PA12 GPIO port.

0: No remap (PA12)

1: Remap (PA10)

*Note: If the PINMUX4[1:0] bitfield of the SYSCFG\_CFGR3 register is at 00, PA12\_RMP must be kept at 0 to prevent conflict due to two GPIO outputs with different output levels connected to the same pin.*

Bit 3 **PA11\_RMP**: PA11 pin remapping

This bit is set and cleared by software. When set, it remaps the PA11 pin to operate as PA9 GPIO port, instead as PA11 GPIO port.

0: No remap (PA11)

1: Remap (PA9)

*Note: If the PINMUX2[1:0] bitfield of the SYSCFG\_CFGR3 register is at 00, PA11\_RMP must be kept at 0 to prevent conflict due to two GPIO outputs with different output levels connected to the same pin.*

Bit 2 Reserved, must be kept at reset value.

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

This bitfield controlled by software selects the memory internally mapped at the address 0x0000 0000. Its reset value is determined by the boot mode configuration. Refer to [Section 2.5: Boot configuration](#) for more details.

x0: Main Flash memory

01: System Flash memory

11: Embedded SRAM

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

Address offset: 0x18

System 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.	LOCKUP_LOCK														
															rw

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

Bit 0 **LOCKUP\_LOCK**: Cortex®-M0+ LOCKUP enable

This bit is set by software and cleared by system reset. When set, it enables the connection of Cortex®-M0+ LOCKUP (HardFault) output to the TIM1/16/17 Break input.

0: Disable

1: Enable

### 7.1.3 SYSCFG configuration register 3 (SYSCFG\_CFGR3)

When several GPIOs are internally connected to the same pin, this register allows assigning the one to keep setting specified by its corresponding GPIOx\_MODER register. The other GPIOs are forced into digital input mode regardless their corresponding GPIOx\_MODER register settings.

This is only effective when the SECURE\_MUXING\_EN bit of the *FLASH option register (FLASH\_OPTR)* is set (default).

When SECURE\_MUXING\_EN is cleared, SYSCFG\_CFGR3 has no effect. All GPIOs connected to the same pin keep the configuration set in their corresponding GPIOx\_MODER register. The user software must ensure that there is no conflict between the GPIOs.

Address offset: 0x3C

System 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.	PINMUX5[1:0]	PINMUX4[1:0]	PINMUX3[1:0]	PINMUX2[1:0]	PINMUX1[1:0]	PINMUX0[1:0]						
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 11:10 **PINMUX5[1:0]**: Pin GPIO multiplexer 5

This bit is set by software and cleared by system reset. It assigns a GPIO to a pin.

Condition: STM32C011x - GPIO assigned to WLCSP12 pin F1

00: PA3

01: PA4

10: PA5

11: PA6

**Bits 9:8 PINMUX4[1:0]: Pin GPIO multiplexer 4**

This bit is set by software and cleared by system reset. It assigns a GPIO to a pin.

Condition: STM32C011x - GPIO assigned to WLCSP12 pin E2

00: PA7

01: PA12

1x: Reserved

*Note: The PA12\_RMP bit of the SYSCFG\_CFGR1 takes priority over the selection through this bitfield. Refer to the description of the SYSCFG\_CFGR1 register for more details.*

**Bits 7:6 PINMUX3[1:0]: Pin GPIO multiplexer 3**

This bit is set by software and cleared by system reset. It assigns a GPIO to a pin.

Condition: STM32C011x - GPIO assigned to SO8 pin 8

00: PA14

01: PB6

10: PC15

11: Reserved

**Bits 5:4 PINMUX2[1:0]: Pin GPIO multiplexer 2**

This bit is set by software and cleared by system reset. It assigns a GPIO to a pin.

Condition: STM32C011x - GPIO assigned to SO8 pin 5

00: PA8

01: PA11

1x: Reserved

*Note: The PA11\_RMP bit of the SYSCFG\_CFGR1 takes priority over the selection through this bitfield. Refer to the description of the SYSCFG\_CFGR1 register for more details.*

**Bits 3:2 PINMUX1[1:0]: Pin GPIO multiplexer 1**

This bit is set by software and cleared by system reset. It assigns a GPIO to a pin.

Condition: STM32C011x - GPIO assigned to SO8 pin 4

00: PF2

01: PA0

10: PA1

11: PA2

**Bits 1:0 PINMUX0[1:0]: Pin GPIO multiplexer 0**

This bit is set by software and cleared by system reset. It assigns a GPIO to a pin.

Condition: STM32C011x - GPIO assigned to SO8 pin 1

00: PB7

01: PC14

1x: Reserved

## 7.1.4 SYSCFG interrupt line 0 status register (SYSCFG\_ITLINE0)

A dedicated set of registers is implemented on the device to collect all pending interrupt sources associated with each interrupt line into a single register. This allows users to check by single read which peripheral requires service in case more than one source is associated to the interrupt line.

All bits in those registers are read only, set by hardware when there is corresponding interrupt request pending and cleared by resetting the interrupt source flags in the peripheral registers.

Address offset: 0x80

System 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.	WWDG														
															r

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

Bit 0 **WWDG**: Window watchdog interrupt pending flag

### 7.1.5 SYSCFG interrupt line 2 status register (SYSCFG\_ITLINE2)

Address offset: 0x88

System 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.	RTC	Res.													
															r

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

Bit 1 **RTC**: RTC interrupt request pending (EXTI line 19)

Bit 0 Reserved, must be kept at reset value.

### 7.1.6 SYSCFG interrupt line 3 status register (SYSCFG\_ITLINE3)

Address offset: 0x8C

System 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.	FLASH ITF	Res.													
															r

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

Bit 1 **FLASH\_ITF**: Flash interface interrupt request pending

Bit 0 Reserved, must be kept at reset value.

### 7.1.7 SYSCFG interrupt line 4 status register (SYSCFG\_ITLINE4)

Address offset: 0x90

System 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.	RCC														
															r

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

Bit 0 **RCC**: Reset and clock control interrupt request pending

### 7.1.8 SYSCFG interrupt line 5 status register (SYSCFG\_ITLINE5)

Address offset: 0x94

System 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.	EXTI1	EXTI0													
														r	r

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

Bit 1 **EXTI1**: EXTI line 1 interrupt request pending

Bit 0 **EXTI0**: EXTI line 0 interrupt request pending

### 7.1.9 SYSCFG interrupt line 6 status register (SYSCFG\_ITLINE6)

Address offset: 0x98

System 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.	EXTI3	EXTI2													
														r	r

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

Bit 1 **EXTI3**: EXTI line 3 interrupt request pending

Bit 0 **EXTI2**: EXTI line 2 interrupt request pending

### 7.1.10 SYSCFG interrupt line 7 status register (SYSCFG\_ITLINE7)

Address offset: 0x9C

System 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.	EXTI15	EXTI14	EXTI13	EXTI12	EXTI11	EXTI10	EXTI9	EXTI8	EXTI7	EXTI6	EXTI5	EXTI4
				r	r	r	r	r	r	r	r	r	r	r	r

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

- Bit 11 **EXTI15**: EXTI line 15 interrupt request pending
- Bit 10 **EXTI14**: EXTI line 14 interrupt request pending
- Bit 9 **EXTI13**: EXTI line 13 interrupt request pending
- Bit 8 **EXTI12**: EXTI line 12 interrupt request pending
- Bit 7 **EXTI11**: EXTI line 11 interrupt request pending
- Bit 6 **EXTI10**: EXTI line 10 interrupt request pending
- Bit 5 **EXTI9**: EXTI line 9 interrupt request pending
- Bit 4 **EXTI8**: EXTI line 8 interrupt request pending
- Bit 3 **EXTI7**: EXTI line 7 interrupt request pending
- Bit 2 **EXTI6**: EXTI line 6 interrupt request pending
- Bit 1 **EXTI5**: EXTI line 5 interrupt request pending
- Bit 0 **EXTI4**: EXTI line 4 interrupt request pending

### 7.1.11 SYSCFG interrupt line 9 status register (SYSCFG\_ITLINE9)

Address offset: 0xA4

System 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.	DMA1_CH1														
															r

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

- Bit 0 **DMA1\_CH1**: DMA1 channel 1 interrupt request pending

### 7.1.12 SYSCFG interrupt line 10 status register (SYSCFG\_ITLINE10)

Address offset: 0xA8

System 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.	DMA1_CH3 DMA1_CH2														
															r r

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

Bit 1 **DMA1\_CH3**: DMA1 channel 3 interrupt request pending

Bit 0 **DMA1\_CH2**: DMA1 channel 2 interrupt request pending

### 7.1.13 SYSCFG interrupt line 11 status register (SYSCFG\_ITLINE11)

Address offset: 0xAC

System 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.	DMAMUX														
															r

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

Bit 0 **DMAMUX**: DMAMUX interrupt request pending

### 7.1.14 SYSCFG interrupt line 12 status register (SYSCFG\_ITLINE12)

Address offset: 0xB0

System 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.	ADC														
															r

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

Bit 0 **ADC**: ADC interrupt request pending

### 7.1.15 SYSCFG interrupt line 13 status register (SYSCFG\_ITLINE13)

Address offset: 0xB4

System 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.	TIM1_BRK														
															r
Res.	TIM1_UPD														
															r
Res.	TIM1_TRG														
															r
Res.	TIM1_CCU														
															r

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

Bit 3 **TIM1\_BRK**: Timer 1 break interrupt request pending

Bit 2 **TIM1\_UPD**: Timer 1 update interrupt request pending

Bit 1 **TIM1\_TRG**: Timer 1 trigger interrupt request pending

Bit 0 **TIM1\_CCU**: Timer 1 commutation interrupt request pending

### 7.1.16 SYSCFG interrupt line 14 status register (SYSCFG\_ITLINE14)

Address offset: 0xB8

System 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.	TIM1_CC														
															r

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

Bit 0 **TIM1\_CC**: Timer 1 capture compare interrupt request pending

### 7.1.17 SYSCFG interrupt line 16 status register (SYSCFG\_ITLINE16)

Address offset: 0xC0

System 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.	TIM3														
															r

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

Bit 0 **TIM3**: Timer 3 interrupt request pending

### 7.1.18 SYSCFG interrupt line 19 status register (SYSCFG\_ITLINE19)

Address offset: 0xCC

System 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.	TIM14														
															r

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

Bit 0 **TIM14**: Timer 14 interrupt request pending

### 7.1.19 SYSCFG interrupt line 21 status register (SYSCFG\_ITLINE21)

Address offset: 0xD4

System 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.	TIM16														
															r

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

Bit 0 **TIM16**: Timer 16 interrupt request pending

### 7.1.20 SYSCFG interrupt line 22 status register (SYSCFG\_ITLINE22)

Address offset: 0xD8

System 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.	TIM17														
															r

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

Bit 0 **TIM17**: Timer 17 interrupt request pending

### 7.1.21 SYSCFG interrupt line 23 status register (SYSCFG\_ITLINE23)

Address offset: 0xDC

System 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.	I2C1														
															r

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

Bit 0 **I2C1**: I2C1 interrupt request pending, combined with EXTI line 23

### 7.1.22 SYSCFG interrupt line 25 status register (SYSCFG\_ITLINE25)

Address offset: 0xE4

System 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.	SPI1														
															r

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

Bit 0 **SPI1**: SPI1 interrupt request pending

### 7.1.23 SYSCFG interrupt line 27 status register (SYSCFG\_ITLINE27)

Address offset: 0xEC

System 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.	USART 1														
															r

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

Bit 0 **USART1**: USART1 interrupt request pending, combined with EXTI line 25

### 7.1.24 SYSCFG interrupt line 28 status register (SYSCFG\_ITLINE28)

Address offset: 0xF0

System reset value: 0x0000 0000

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

Bit 0 **USART2**: USART2 interrupt request pending (EXTI line 26)

### 7.1.25 SYSCFG register map

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

**Table 26. SYSCFG register map and reset values**

**Table 26. SYSCFG register map and reset values (continued)**

**Table 26. SYSCFG register map and reset values (continued)**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xDC	<b>SYSCFG_ITLINE23</b>	Res.	o I2C1	o SPI1																													
	Reset value																																
0xE0	<b>Reserved</b>																																
0xE4	<b>SYSCFG_ITLINE25</b>	Res.	o USART1																														
	Reset value																																
0xE8	<b>Reserved</b>																																
	Reset value																																
0xEC	<b>SYSCFG_ITLINE27</b>	Res.	o USART2																														
	Reset value																																
0xF0	<b>SYSCFG_ITLINE28</b>	Res.	o I2C1																														
	Reset value																																

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 8 Interconnect matrix

### 8.1 Introduction

Several peripherals have direct connections between them.

This allows autonomous communication and/or synchronization between peripherals, saving CPU resources thus power consumption.

In addition, these hardware connections remove software latency and allow design of predictable systems.

Depending on peripherals, these interconnections can operate in Run, Sleep, and Stop mode.

For availability of peripherals on different STM32C0x1 products, refer to [Section 1.4: Availability of peripherals](#).

### 8.2 Connection summary

**Table 27. Interconnect matrix<sup>(1)(2)</sup>**

Source	Destination							
	TIM1	TIM3	TIM14	TIM16	TIM17	ADC1	DMAMUX	IRTIM
TIM1	-	8.3.1	-	-	-	8.3.2	-	-
TIM3	8.3.1	-	-	-	-	8.3.2	-	-
TIM14	-	8.3.1	-	-	-	-	8.3.8	-
TIM16	-	-	-	-	-	-	-	8.3.7
TIM17	8.3.1	-	-	-	-	-	-	8.3.7
USART1	-	-	-	-	-	-	-	8.3.7
USART2	-	-	-	-	-	-	-	8.3.7
ADC	8.3.3	-	-	-	-	-	-	-
Temp. sensor	-	-	-	-	-	8.3.5	-	-
VREFINT	-	-	-	-	-	8.3.5	-	-
HSE	-	-	8.3.4	-	8.3.4	-	-	-
LSE	-	-	-	8.3.4	-	-	-	-
LSI	-	-	-	8.3.4	-	-	-	-
MCO	-	-	8.3.4	-	8.3.4	-	-	-
MCO2	-	-	8.3.4	8.3.4	8.3.4	-	-	-
EXTI	-	-	-	-	-	8.3.2	8.3.2	-

**Table 27. Interconnect matrix<sup>(1)(2)</sup> (continued)**

Source	Destination							
	TIM1	TIM3	TIM14	TIM16	TIM17	ADC1	DMAMUX	IRTIM
RTC	-	-	8.3.4	-	-	-	-	-
SYST ERR	8.3.6	-	-	8.3.6	8.3.6	-	-	-

1. Numbers in the table are links to corresponding sub-sections in [Section 8.3: Interconnection details](#).
2. The “-” symbol in grayed cells means “no interconnection”.

## 8.3 Interconnection details

### 8.3.1 From TIM1, TIM3, TIM14, and TIM17, to TIM1 and TIM3

#### Purpose

Some of the TIMx timers are linked together internally for timer synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop or clock the counter of another timer configured in slave mode.

A description of the feature is provided in: [Section 16.3.19: Timer synchronization](#).

The modes of synchronization are detailed in:

- [Section 15.3.26: Timer synchronization](#) for advanced-control timer TIM1
- [Section 16.3.18: Timers and external trigger synchronization](#) for general-purpose timer TIM3

#### Triggering signals

The output (from master) is on signal TIMx\_TRGO (and TIMx\_TRGOx), following a configurable timer event.

With TIM14 and TIM17 timers that do not have a trigger output, the output compare 1 is used instead.

The input (to slave) is on signals TIMx\_ITR0/ITR1/ITR2/ITR3.

The input and output signals for TIM1 are shown in [Figure 55: Advanced-control timer block diagram](#).

The possible master/slave connections are given in [Table 65: TIM1 internal trigger connection](#).

#### Relevant power modes

These interconnections operate in Run and Sleep modes.

### 8.3.2 From TIM1, TIM3, and EXTI, to ADC

#### Purpose

The general-purpose timer TIM3, advanced-control timer TIM1, and EXTI can be used to generate an ADC triggering event.

TIMx synchronization is described in: [Section 15.3.27: ADC synchronization](#).

ADC synchronization is described in: [Section 14.5: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN\)](#).

#### Triggering signals

The output (from timer) is on signal TIMx\_TRGO, TIMx\_TRGO2 or TIMx\_CCx event.

The input (to ADC) is on signal TRG[7:0].

The connection between timers and ADC is provided in [Table 53: External triggers](#).

#### Relevant power modes

These interconnections operate in Run and Sleep modes.

### 8.3.3 From ADC to TIM1

#### Purpose

ADC can provide trigger event through watchdog signals to the advanced-control timer TIM1.

A description of the ADC analog watchdog setting is provided in: [Section 14.8: Analog window watchdogs](#).

Trigger settings on the timer are provided in: [Section 15.3.4: External trigger input](#).

#### Triggering signals

The output (from ADC) is on signals ADC\_AWD $x$ \_OUT  $x = 1, 2, 3$  (three watchdogs per ADC) and the input (to timer) on signal TIMx\_ETR (external trigger).

#### Relevant power modes

This interconnection operates in Run and Sleep modes.

### 8.3.4 From HSE, LSE, LSI, MCO, MCO2, and RTC, to TIM14, TIM16, and TIM17

#### Purpose

External clocks (HSE, LSE), internal clock (LSI), microcontroller output clock (MCO and MCO2), RTC clock, and GPIO can be selected as inputs to capture channel 1 of some of TIM14/16/TIM17 timers.

The timers allow calibrating or precisely measuring internal clocks such as HSI48 or LSI, using accurate clocks such as LSE or HSE/32 for timing reference. See details in [Section 5.2.13: Internal/external clock measurement with TIM14/TIM16/TIM17](#).

When low-speed external (LSE) oscillator is used, no additional hardware connections are required.

#### Relevant power modes

These interconnections operate in Run and Sleep modes.

### 8.3.5 From internal analog sources to ADC

#### Purpose

The internal temperature sensor output voltage  $V_{TS}$  and the internal reference voltage  $V_{REFINT}$  channel are connected to ADC input channels.

More information is in:

- [Section 14.2: ADC main features](#)
- [Section 14.4.8: Channel selection \(CHSEL, SCANDIR, CHSELRMOD\)](#)
- [Figure 14.10: Temperature sensor and internal reference voltage](#)

#### Relevant power modes

These interconnections operate in Run and Sleep modes.

### 8.3.6 From system errors to TIM1, TIM16, and TIM17

#### Purpose

CSS, CPU HardFault, and RAM parity error can generate system errors in the form of timer break toward TIM1, TIM16, and TIM17.

The purpose of the break function is to protect power switches driven by PWM signals from the timers.

The relevant information is in:

- [Section 15.3.16: Using the break function \(TIM1\)](#)
- [Section 18.3.11: Using the break function \(TIM16/TIM17\)](#)
- [Figure 180: TIM16/TIM17 block diagram](#)

#### Relevant power modes

These interconnections operate in Run and Sleep modes.

### 8.3.7 From TIM16, TIM17, USART1, and USART2, to IRTIM

#### Purpose

TIMx\_OC1 output channel of TIM17 timer, associated with USART1 or USART2 transmission signal, can generate the infrared output waveform.

The functionality is described in [Section 19: Infrared interface \(IRTIM\)](#).

#### Relevant power modes

These interconnections operate in Run and Sleep modes.

### 8.3.8 From TIM14 and EXTI to DMAMUX

#### Purpose

TIM14 general-purpose timer and EXTI can be used as triggering event to DMAMUX.

#### Relevant power modes

These interconnections operate in Run and Sleep modes.

## 9 Direct memory access controller (DMA)

### 9.1 Introduction

The direct memory access (DMA) controller is a bus master and system peripheral.

The DMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories, upon the control of an off-loaded CPU.

The DMA controller features a single AHB master architecture.

There is one instance of DMA with 3 channels.

Each channel is dedicated to managing memory access requests from one or more peripherals. The DMA includes an arbiter for handling the priority between DMA requests.

### 9.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (4 levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to  $2^{16} - 1$
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.

## 9.3 DMA implementation

### 9.3.1 DMA1

DMA1 is implemented with the hardware configuration parameters shown in the table below.

**Table 28. DMA implementation**

Feature	DMA
Number of channels	3

### 9.3.2 DMA request mapping

The DMA controller is connected to DMA requests from the AHB/APB peripherals through the DMAMUX peripheral.

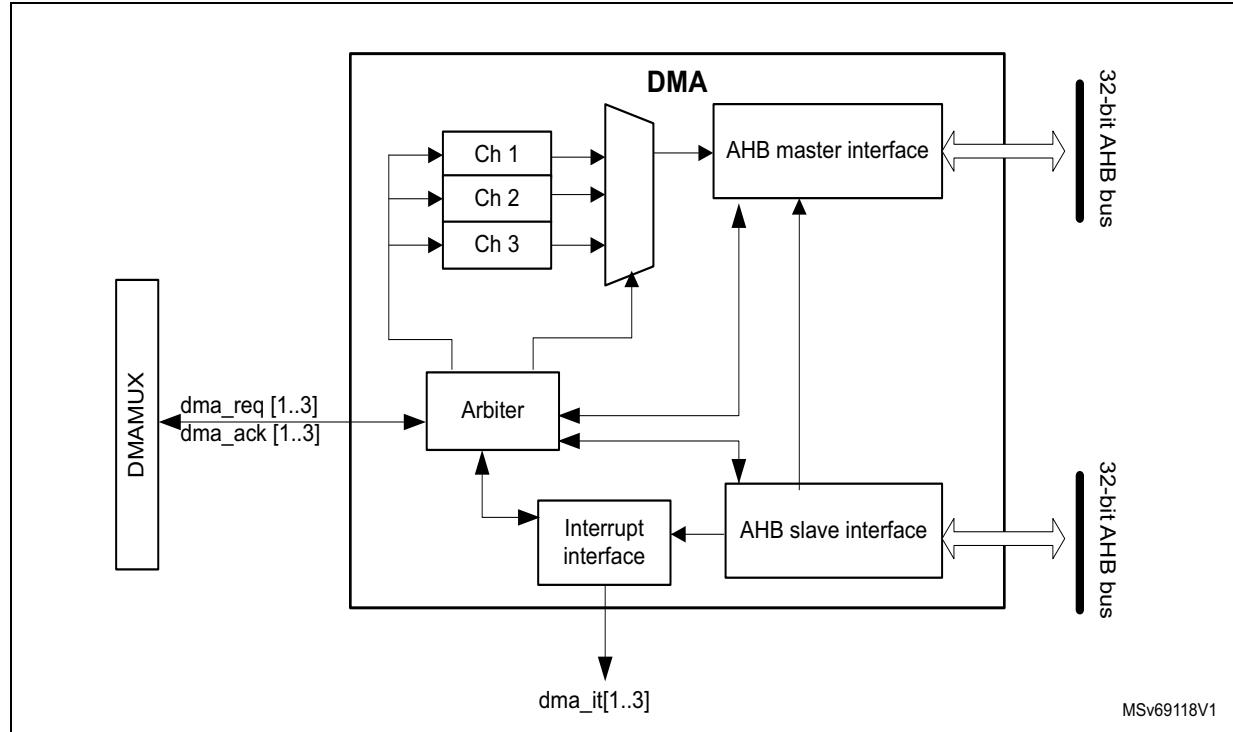
For the mapping of the different requests, refer to the DMAMUX section.

## 9.4 DMA functional description

### 9.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.

**Figure 19. DMA block diagram**



The DMA controller performs direct memory transfer by sharing the AHB system bus with other system masters. The bus matrix implements round-robin scheduling. DMA requests may stop the CPU access to the system bus for a number of bus cycles, when CPU and DMA target the same destination (memory or peripheral).

According to its configuration through the AHB slave interface, the DMA controller arbitrates between the DMA channels and their associated received requests. The DMA controller also schedules the DMA data transfers over the single AHB port master.

The DMA controller generates an interrupt per channel to the interrupt controller.

#### 9.4.2 DMA pins and internal signals

**Table 29. DMA internal input/output signals**

Signal name	Signal type	Description
dma_req[x]	Input	DMA channel x request
dma_ack[x]	Output	DMA channel x acknowledge
dma_it[x]	Output	DMA channel x interrupt

#### 9.4.3 DMA transfers

The software configures the DMA controller at channel level, in order to perform a block transfer, composed of a sequence of AHB bus transfers.

A DMA block transfer may be requested from a peripheral, or triggered by the software in case of memory-to-memory transfer.

After an event, the following steps of a single DMA transfer occur:

1. The peripheral sends a single DMA request signal to the DMA controller.
2. The DMA controller serves the request, depending on the priority of the channel associated to this peripheral request.
3. As soon as the DMA controller grants the peripheral, an acknowledge is sent to the peripheral by the DMA controller.
4. The peripheral releases its request as soon as it gets the acknowledge from the DMA controller.
5. Once the request is de-asserted by the peripheral, the DMA controller releases the acknowledge.

The peripheral may order a further single request and initiate another single DMA transfer.

The request/acknowledge protocol is used when a peripheral is either the source or the destination of the transfer. For example, in case of memory-to-peripheral transfer, the peripheral initiates the transfer by driving its single request signal to the DMA controller. The DMA controller reads then a single data in the memory and writes this data to the peripheral.

For a given channel x, a DMA block transfer consists of a repeated sequence of:

- a single DMA transfer, encapsulating two AHB transfers of a single data, over the DMA AHB bus master:
  - a single data read (byte, half-word or word) from the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.

The start address used for the first single transfer is the base address of the peripheral or memory, and is programmed in the DMA\_CPARx or DMA\_CMARx register.

- a single data write (byte, half-word or word) to the peripheral data register or a location in the memory, addressed through an internal current peripheral/memory address register.
- The start address used for the first transfer is the base address of the peripheral or memory, and is programmed in the DMA\_CPARx or DMA\_CMARx register.
- post-decrementing of the programmed DMA\_CNDTRx register  
This register contains the remaining number of data items to transfer (number of AHB ‘read followed by write’ transfers).

This sequence is repeated until DMA\_CNDTRx is null.

*Note:* *The AHB master bus source/destination address must be aligned with the programmed size of the transferred single data to the source/destination.*

#### 9.4.4 DMA arbitration

The DMA arbiter manages the priority between the different channels.

When an active channel x is granted by the arbiter (hardware requested or software triggered), a single DMA transfer is issued (such as a AHB ‘read followed by write’ transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the DMA\_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low
- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 1 gets priority over channel 2.

When a channel x is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single DMA transfer of this channel x. Whenever there is another concurrent active requested channel, the DMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

#### 9.4.5 DMA channels

Each channel may handle a DMA transfer between a peripheral register located at a fixed address, and a memory address. The amount of data items to transfer is programmable. The register that contains the amount of data items to transfer is decremented after each transfer.

A DMA channel is programmed at block transfer level.

## Programmable data sizes

The transfer sizes of a single data (byte, half-word, or word) to the peripheral and memory are programmable through, respectively, the PSIZE[1:0] and MSIZE[1:0] fields of the DMA\_CCRx register.

## Pointer incrementation

The peripheral and memory pointers may be automatically incremented after each transfer, depending on the PINC and MINC bits of the DMA\_CCRx register.

If the **incremented mode** is enabled (PINC or MINC set to 1), the address of the next transfer is the address of the previous one incremented by 1, 2 or 4, depending on the data size defined in PSIZE[1:0] or MSIZE[1:0]. The first transfer address is the one programmed in the DMA\_CPARx or DMA\_CMARx register. During transfers, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel x is configured in **non-circular mode**, no DMA request is served after the last data transfer (once the number of single data to transfer reaches zero). The DMA channel must be disabled in order to reload a new number of data items into the DMA\_CNDTRx register.

**Note:** *If the channel x is disabled, the DMA registers are not reset. The DMA channel registers (DMA\_CCRx, DMA\_CPARx and DMA\_CMARx) retain the initial values programmed during the channel configuration phase.*

In **circular mode**, after the last data transfer, the DMA\_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA\_CPARx and DMA\_CMARx registers.

## Channel configuration procedure

The following sequence is needed to configure a DMA channel x:

1. Set the peripheral register address in the DMA\_CPARx register.  
The data is moved from/to this address to/from the memory after the peripheral event, or after the channel is enabled in memory-to-memory mode.
2. Set the memory address in the DMA\_CMARx register.  
The data is written to/read from the memory after the peripheral event or after the channel is enabled in memory-to-memory mode.
3. Configure the total number of data to transfer in the DMA\_CNDTRx register.  
After each data transfer, this value is decremented.
4. Configure the parameters listed below in the DMA\_CCRx register:
  - the channel priority
  - the data transfer direction
  - the circular mode
  - the peripheral and memory incremented mode
  - the peripheral and memory data size
  - the interrupt enable at half and/or full transfer and/or transfer error
5. Activate the channel by setting the EN bit in the DMA\_CCRx register.

A channel, as soon as enabled, may serve any DMA request from the peripheral connected to this channel, or may start a memory-to-memory block transfer.

**Note:** *The two last steps of the channel configuration procedure may be merged into a single access to the DMA\_CCRx register, to configure and enable the channel.*

### Channel state and disabling a channel

A channel x in active state is an enabled channel (read DMA\_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA\_CCRx.EN set to 1) and afterwards with no occurred transfer error (DMA\_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (DMA\_CCRx.EN = 0).

The three following use cases may happen:

- Suspend and resume a channel

This corresponds to the two following actions:

- An active channel is disabled by software (writing DMA\_CCRx.EN = 0 whereas DMA\_CCRx.EN = 1).
- The software enables the channel again (DMA\_CCRx.EN set to 1) without reconfiguring the other channel registers (such as DMA\_CNDTRx, DMA\_CPARx and DMA\_CMARx).

This case is not supported by the DMA hardware, that does not guarantee that the remaining data transfers are performed correctly.

- Stop and abort a channel

If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the DMA\_CNDTRx register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- Abort and restart a channel

This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.

This is supported by the hardware if the following conditions are met:

- The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.
- The software must operate separated write accesses to the same DMA\_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the DMA\_CCRx if a configuration change is needed. There are read-only DMA\_CCRx register fields when DMA\_CCRx.EN=1. Finally enable again the channel.

When a channel transfer error occurs, the EN bit of the DMA\_CCRx register is cleared by hardware. This EN bit can not be set again by software to re-activate the channel x, until the TEIFx bit of the DMA\_ISR register is set.

### Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the DMA\_CCRx register.

**Note:**

*The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the DMA\_CCRx register. When the circular mode is activated, the amount of data to transfer is*

*automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.*

*In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel.*

*The software must explicitly program the DMA\_CNDTRx value before starting/enabling a transfer, and after having stopped a circular transfer.*

### Memory-to-memory mode

The DMA channels may operate without being triggered by a request from a peripheral. This mode is called memory-to-memory mode, and is initiated by software.

If the MEM2MEM bit in the DMA\_CCRx register is set, the channel, if enabled, initiates transfers. The transfer stops once the DMA\_CNDTRx register reaches zero.

**Note:** *The memory-to-memory mode must not be used in circular mode. Before enabling a channel in memory-to-memory mode (MEM2MEM = 1), the software must clear the CIRC bit of the DMA\_CCRx register.*

### Peripheral-to-peripheral mode

Any DMA channel can operate in peripheral-to-peripheral mode:

- when the hardware request from a peripheral is selected to trigger the DMA channel  
This peripheral is the DMA initiator and paces the data transfer from/to this peripheral to/from a register belonging to another memory-mapped peripheral (this one being not configured in DMA mode).
- when no peripheral request is selected and connected to the DMA channel  
The software configures a register-to-register transfer by setting the MEM2MEM bit of the DMA\_CCRx register.

### Programming transfer direction, assigning source/destination

The value of the DIR bit of the DMA\_CCRx register sets the direction of the transfer, and consequently, it identifies the source and the destination, regardless the source/destination type (peripheral or memory):

- **DIR = 1** defines typically a memory-to-peripheral transfer. More generally, if DIR = 1:
  - The **source** attributes are defined by the DMA\_MARx register, the MSIZE[1:0] field and MINC bit of the DMA\_CCRx register.  
Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the source peripheral in peripheral-to-peripheral mode.
  - The **destination** attributes are defined by the DMA\_PARx register, the PSIZE[1:0] field and PINC bit of the DMA\_CCRx register.  
Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the destination memory in memory-to-memory mode.
- **DIR = 0** defines typically a peripheral-to-memory transfer. More generally, if DIR = 0:
  - The **source** attributes are defined by the DMA\_PARx register, the PSIZE[1:0] field and PINC bit of the DMA\_CCRx register.  
Regardless of their usual naming, these ‘peripheral’ register, field and bit are used to define the source memory in memory-to-memory mode
  - The **destination** attributes are defined by the DMA\_MARx register, the MSIZE[1:0] field and MINC bit of the DMA\_CCRx register.

Regardless of their usual naming, these ‘memory’ register, field and bit are used to define the destination peripheral in peripheral-to-peripheral mode.

#### 9.4.6 DMA data width, alignment and endianness

When PSIZE[1:0] and MSIZE[1:0] are not equal, the DMA controller performs some data alignments as described in the table below.

**Table 30. Programmable data width and endian behavior (when PINC = MINC = 1)**

Source port width (MSIZE if DIR = 1, else PSIZE)	Destination port width (PSIZE if DIR = 1, else MSIZE)	Number of data items to transfer (NDT)	Source content: address / data (DMA_CMARx if DIR = 1, else DMA_CPARx)	DMA transfers	Destination content: address / data (DMA_CPARx if DIR = 1, else DMA_CMARx)
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 B2[7:0] @0x2 then write 00B2[15:0] @0x4 4: read B3[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 B2[7:0] @0x2 then write 000000B2[31:0] @0x8 4: read B3[7:0] @0x3 then write 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B0[7:0] @0x0 2: read B3B2[15:0] @0x2 then write B2[7:0] @0x1 3: read B5B4[15:0] @0x4 then write B4[7:0] @0x2 4: read B7B6[15:0] @0x6 then write B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write B1B0[15:0] @0x0 2: read B3B2[15:0] @0x2 then write B3B2[15:0] @0x2 3: read B5B4[15:0] @0x4 then write B5B4[15:0] @0x4 4: read B7B6[15:0] @0x6 then write B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: read B1B0[15:0] @0x0 then write 0000B1B0[31:0] @0x0 2: read B3B2[15:0] @0x2 then write 0000B3B2[31:0] @0x4 3: read B5B4[15:0] @0x4 then write 0000B5B4[31:0] @0x8 4: read B7B6[15:0] @0x6 then write 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B0[7:0] @0x0 2: read B7B6B5B4[31:0] @0x4 then write B4[7:0] @0x1 3: read BBBAB9B8[31:0] @0x8 then write B8[7:0] @0x2 4: read BFBEBDBC[31:0] @0xC then write BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: read B3B2B1B0[31:0] @0x0 then write B1B0[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 AHB peripherals not supporting byte/half-word write transfers

When the DMA controller initiates an AHB byte or half-word write transfer, the data are duplicated on the unused lanes of the AHB master 32-bit data bus (HWDATA[31:0]).

When the AHB slave peripheral does not support byte or half-word write transfers and does not generate any error, the DMA controller writes the 32 HWDATA bits as shown in the two examples below:

- To write the half-word 0xABCD, the DMA controller sets the HWDATA bus to 0xABCDABCD with a half-word data size (HSIZE = HalfWord in AHB master bus).
- To write the byte 0xAB, the DMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2 or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses, is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

#### 9.4.7 DMA error management

A DMA transfer error is generated when reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or write access, the faulty channel x is automatically disabled through a hardware clear of its EN bit in the corresponding DMA\_CCRx register.

The TEIFx bit of the DMA\_ISR register is set. An interrupt is then generated if the TEIE bit of the DMA\_CCRx register is set.

The EN bit of the DMA\_CCRx register can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA\_ISR register is cleared (by setting the CTEIFx bit of the DMA\_IFCR register).

When the software is notified with a transfer error over a channel which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both DMA and the peripheral in DMA mode for a new transfer.

## 9.5 DMA interrupts

An interrupt can be generated on a half transfer, transfer complete or transfer error for each DMA channel x. Separate interrupt enable bits are available for flexibility.

**Table 31. DMA interrupt requests**

Interrupt request	Interrupt event	Event flag	Interrupt enable bit
Channel x interrupt	Half transfer on channel x	HTIFx	HTIEx
	Transfer complete on channel x	TCIFx	TCIEx
	Transfer error on channel x	TEIFx	TEIEx
	Half transfer or transfer complete or transfer error on channel x	GIFx	-

## 9.6 DMA registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

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

### 9.6.1 DMA interrupt status register (DMA\_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

Every status bit is cleared by hardware when the software sets the corresponding clear bit or the corresponding global clear bit CGIFx, in the DMA\_IFCR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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.	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 11 **TEIF3**: Transfer error (TE) flag for channel 3

- 0: No TE event
- 1: A TE event occurred.

Bit 10 **HTIF3**: Half transfer (HT) flag for channel 3

- 0: No HT event
- 1: An HT event occurred.

Bit 9 **TCIF3**: Transfer complete (TC) flag for channel 3

- 0: No TC event
- 1: A TC event occurred.

Bit 8 **GIF3**: Global interrupt flag for channel 3

- 0: No TE, HT, or TC event
- 1: A TE, HT, or TC event occurred.

- Bit 7 **TEIF2**: Transfer error (TE) flag for channel 2  
 0: No TE event  
 1: A TE event occurred.
- Bit 6 **HTIF2**: Half transfer (HT) flag for channel 2  
 0: No HT event  
 1: An HT event occurred.
- Bit 5 **TCIF2**: Transfer complete (TC) flag for channel 2  
 0: No TC event  
 1: A TC event occurred.
- Bit 4 **GIF2**: Global interrupt flag for channel 2  
 0: No TE, HT, or TC event  
 1: A TE, HT, or TC event occurred.
- Bit 3 **TEIF1**: Transfer error (TE) flag for channel 1  
 0: No TE event  
 1: A TE event occurred.
- Bit 2 **HTIF1**: Half transfer (HT) flag for channel 1  
 0: No HT event  
 1: An HT event occurred.
- Bit 1 **TCIF1**: Transfer complete (TC) flag for channel 1  
 0: No TC event  
 1: A TC event occurred.
- Bit 0 **GIF1**: Global interrupt flag for channel 1  
 0: No TE, HT, or TC event  
 1: A TE, HT, or TC event occurred.

## 9.6.2 DMA interrupt flag clear register (DMA\_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

Setting the global clear bit CGIFx of the channel x in this DMA\_IFCR register, causes the DMA hardware to clear the corresponding GIFx bit and any individual flag among TEIFx, HTIFx, TCIFx, in the DMA\_ISR register.

Setting any individual clear bit among CTEIFx, CHTIFx, CTCIFx in this DMA\_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIFx in the DMA\_ISR register, provided that none of the two other individual flags is set.

Writing 0 into any flag clear bit has no effect.

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.	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
				w	w	w	w	w	w	w	w	w	w	w	w

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

- Bit 11 **CTEIF3**: Transfer error flag clear for channel 3
- Bit 10 **CHTIF3**: Half transfer flag clear for channel 3
- Bit 9 **CTCIF3**: Transfer complete flag clear for channel 3
- Bit 8 **CGIF3**: Global interrupt flag clear for channel 3
- Bit 7 **CTEIF2**: Transfer error flag clear for channel 2
- Bit 6 **CHTIF2**: Half transfer flag clear for channel 2
- Bit 5 **CTCIF2**: Transfer complete flag clear for channel 2
- Bit 4 **CGIF2**: Global interrupt flag clear for channel 2
- Bit 3 **CTEIF1**: Transfer error flag clear for channel 1
- Bit 2 **CHTIF1**: Half transfer flag clear for channel 1
- Bit 1 **CTCIF1**: Transfer complete flag clear for channel 1
- Bit 0 **CGIF1**: Global interrupt flag clear for channel 1

### 9.6.3 DMA channel x configuration register (DMA\_CCRx)

Address offset:  $0x08 + 0x14 * (x - 1)$ , ( $x = 1$  to  $3$ )

Reset value: 0x0000 0000

The register fields/bits MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

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
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **MEM2MEM**: Memory-to-memory mode

0: Disabled

1: Enabled

*Note:* This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).

Bits 13:12 **PL[1:0]**: Priority level

00: Low

01: Medium

10: High

11: Very high

*Note:* This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).

**Bits 11:10 MSIZE[1:0]: Memory size**

Defines the data size of each DMA transfer to the identified memory.

In memory-to-memory mode, this bitfield identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: Reserved

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

**Bits 9:8 PSIZE[1:0]: Peripheral size**

Defines the data size of each DMA transfer to the identified peripheral.

In memory-to-memory mode, this bitfield identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

00: 8 bits

01: 16 bits

10: 32 bits

11: Reserved

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

**Bit 7 MINC: Memory increment mode**

Defines the increment mode for each DMA transfer to the identified memory.

In memory-to-memory mode, this bit identifies the memory source if DIR = 1 and the memory destination if DIR = 0.

In peripheral-to-peripheral mode, this bit identifies the peripheral source if DIR = 1 and the peripheral destination if DIR = 0.

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

**Bit 6 PINC: Peripheral increment mode**

Defines the increment mode for each DMA transfer to the identified peripheral.

In memory-to-memory mode, this bit identifies the memory destination if DIR = 1 and the memory source if DIR = 0.

In peripheral-to-peripheral mode, this bit identifies the peripheral destination if DIR = 1 and the peripheral source if DIR = 0.

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

**Bit 5 CIRC: Circular mode**

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

**Bit 4 DIR:** Data transfer direction

This bit must be set only in memory-to-peripheral and peripheral-to-memory modes.

0: Read from peripheral

- Source attributes are defined by PSIZE and PINC, plus the DMA\_CPARx register.  
This is still valid in a memory-to-memory mode.

- Destination attributes are defined by MSIZE and MINC, plus the DMA\_CMARx register. This is still valid in a peripheral-to-peripheral mode.

1: Read from memory

- Destination attributes are defined by PSIZE and PINC, plus the DMA\_CPARx register. This is still valid in a memory-to-memory mode.

- Source attributes are defined by MSIZE and MINC, plus the DMA\_CMARx register.  
This is still valid in a peripheral-to-peripheral mode.

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

**Bit 3 TEIE:** Transfer error interrupt enable

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

**Bit 2 HTIE:** Half transfer interrupt enable

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

**Bit 1 TCIE:** Transfer complete interrupt enable

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

**Bit 0 EN:** Channel enable

When a channel transfer error occurs, this bit is cleared by hardware. It can not be set again by software (channel x re-activated) until the TEIFx bit of the DMA\_ISR register is cleared (by setting the CTEIFx bit of the DMA\_IFCR register).

0: Disabled

1: Enabled

*Note: This bit is set and cleared by software.*

### 9.6.4 DMA channel x number of data to transfer register (DMA\_CNDTR<sub>x</sub>)

Address offset: 0x0C + 0x14 \* (x - 1), (x = 1 to 3)

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]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **NDT[15:0]**: Number of data to transfer (0 to  $2^{16} - 1$ )

This bitfield is updated by hardware when the channel is enabled:

- It is decremented after each single DMA ‘read followed by write’ transfer, indicating the remaining amount of data items to transfer.
- It is kept at zero when the programmed amount of data to transfer is reached, if the channel is not in circular mode (CIRC = 0 in the DMA\_CCRx register).
- It is reloaded automatically by the previously programmed value, when the transfer is complete, if the channel is in circular mode (CIRC = 1).

If this bitfield is zero, no transfer can be served whatever the channel status (enabled or not).

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is read-only when the channel is enabled (EN = 1).*

### 9.6.5 DMA channel x peripheral address register (DMA\_CPARx)

Address offset:  $0x10 + 0x14 * (x - 1)$ , (x = 1 to 3)

Reset value: 0x0000 0000

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

Bits 31:0 **PA[31:0]**: Peripheral address

It contains the base address of the peripheral data register from/to which the data is read/written.

When PSIZE[1:0] = 01 (16 bits), bit 0 of PA[31:0] is ignored. Access is automatically aligned to a half-word address.

When PSIZE[1:0] = 10 (32 bits), bits 1 and 0 of PA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this bitfield identifies the memory destination address if DIR = 1 and the memory source address if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral destination address if DIR = 1 and the peripheral source address if DIR = 0.

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

### 9.6.6 DMA channel x memory address register (DMA\_CMARx)

Address offset:  $0x14 + 0x14 * (x - 1)$ , ( $x = 1$  to  $3$ )

Reset value: 0x0000 0000

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]**: Peripheral address

It contains the base address of the memory from/to which the data is read/written.

When MSIZE[1:0] = 01 (16 bits), bit 0 of MA[31:0] is ignored. Access is automatically aligned to a half-word address.

When MSIZE[1:0] = 10 (32 bits), bits 1 and 0 of MA[31:0] are ignored. Access is automatically aligned to a word address.

In memory-to-memory mode, this bitfield identifies the memory source address if DIR = 1 and the memory destination address if DIR = 0.

In peripheral-to-peripheral mode, this bitfield identifies the peripheral source address if DIR = 1 and the peripheral destination address if DIR = 0.

*Note: This bitfield is set and cleared by software. It must not be written when the channel is enabled (EN = 1). It is not read-only when the channel is enabled (EN = 1).*

### 9.6.7 DMA register map

**Table 32. DMA register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x000	DMA_ISR	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	
0x004	DMA_IFCR	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	
0x008	DMA_CCR1	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	
0x00C	DMA_CNDTR1	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	
0x010	DMA_CPAR1	PA[31:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved	Reserved.															

**Table 32. DMA register map and reset values (continued)**

Refer to [Section 2.2](#) for the register boundary addresses.

## 10 DMA request multiplexer (DMAMUX)

### 10.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controller of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in [Section 10.3.1](#).

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend upon product implementation. They are detailed in [Section 10.3.2](#).

## 10.2 DMAMUX main features

- 3-channel programmable DMA request line multiplexer output
- 4-channel DMA request generator
- 23 trigger inputs to DMA request generator
- 23 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - Up to 49 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining

## 10.3 DMAMUX implementation

### 10.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

**Table 33. DMAMUX instantiation**

Feature	DMAMUX
Number of DMAMUX output request channels	3
Number of DMAMUX request generator channels	4
Number of DMAMUX request trigger inputs	23
Number of DMAMUX synchronization inputs	23
Number of DMAMUX peripheral request inputs	49

### 10.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

**Table 34. DMAMUX: assignment of multiplexer inputs to resources**

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux1_req_gen0	19	Reserved	37	TIM3_UP
2	dmamux1_req_gen1	20	TIM1_CH1	38	Reserved

**Table 34. DMAMUX: assignment of multiplexer inputs to resources (continued)**

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
3	dmamux1_req_gen2	21	TIM1_CH2	39	Reserved
4	dmamux1_req_gen3	22	TIM1_CH3	40	Reserved
5	adc_dma	23	TIM1_CH4	41	Reserved
6	Reserved	24	TIM1_TRIG_COM	42	Reserved
7	Reserved	25	TIM1_UP	43	Reserved
8	Reserved	26	Reserved	44	TIM16_CH1
9	Reserved	27	Reserved	45	TIM16_TRIG_COM
10	i2c1_rx_dma	28	Reserved	46	TIM16_UP
11	i2c1_tx_dma	29	Reserved	47	TIM17_CH1
12	Reserved	30	Reserved	48	TIM17_TRIG_COM
13	Reserved	31	Reserved	49	TIM17_UP
14	Reserved	32	TIM3_CH1	50	uart1_rx_dma
15	Reserved	33	TIM3_CH2	51	uart1_tx_dma
16	spi1_rx_dma	34	TIM3_CH3	52	uart2_rx_dma
17	spi1_tx_dma	35	TIM3_CH4	53	uart2_tx_dma
18	Reserved	36	TIM3_TRIG	54	Reserved

**Table 35. DMAMUX: assignment of trigger inputs to resources**

Trigger input	Resource	Trigger input	Resource
0	EXTI LINE0	12	EXTI LINE12
1	EXTI LINE1	13	EXTI LINE13
2	EXTI LINE2	14	EXTI LINE14
3	EXTI LINE3	15	EXTI LINE15
4	EXTI LINE4	16	dmamux_evt0
5	EXTI LINE5	17	dmamux_evt1
6	EXTI LINE6	18	dmamux_evt2
7	EXTI LINE7	19	Reserved
8	EXTI LINE8	20	Reserved
9	EXTI LINE9	21	TIM14
10	EXTI LINE10	22	Reserved
11	EXTI LINE11	23	Reserved

**Table 36. DMAMUX: assignment of synchronization inputs to resources**

Trigger input	Resource	Trigger input	Resource
0	EXTI LINE0	12	EXTI LINE12
1	EXTI LINE1	13	EXTI LINE13

**Table 36. DMAMUX: assignment of synchronization inputs to resources (continued)**

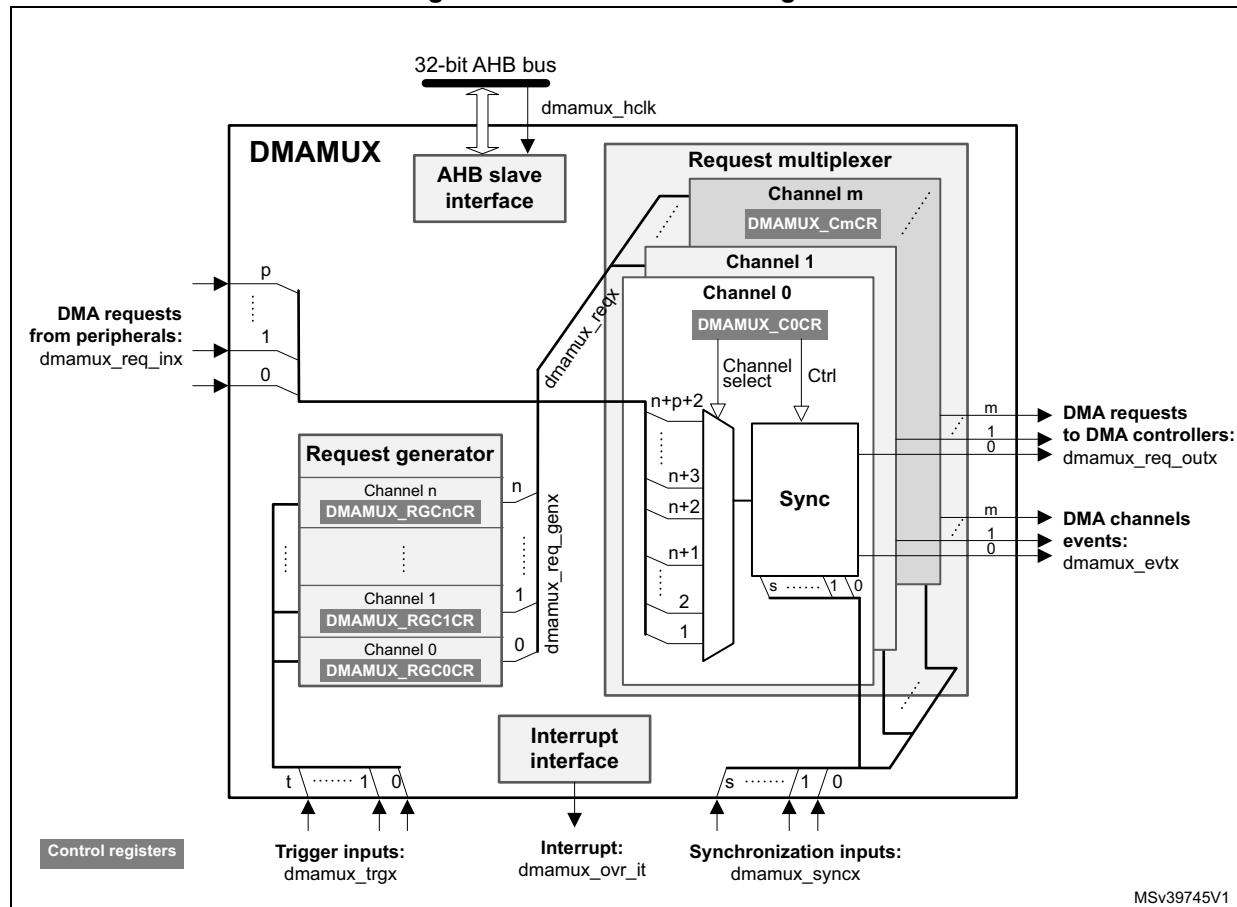
Trigger input	Resource	Trigger input	Resource
2	EXTI LINE2	14	EXTI LINE14
3	EXTI LINE3	15	EXTI LINE15
4	EXTI LINE4	16	dmamux_evt0
5	EXTI LINE5	17	dmamux_evt1
6	EXTI LINE6	18	dmamux_evt2
7	EXTI LINE7	19	Reserved
8	EXTI LINE8	20	Reserved
9	EXTI LINE9	21	TIM14
10	EXTI LINE10	22	Reserved
11	EXTI LINE11	23	Reserved

## 10.4 DMAMUX functional description

### 10.4.1 DMAMUX block diagram

*Figure 20* shows the DMAMUX block diagram.

Figure 20. DMAMUX block diagram



MSv39745V1

DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (**dmamux\_reqx**) from peripherals (**dmamux\_req\_inx**) and from channels of the DMAMUX request generator sub-block (**dmamux\_req\_genx**)
- DMAMUX request outputs to channels of DMA controllers (**dmamux\_req\_outx**)
- Internal or external signals to DMA request trigger inputs (**dmamux\_trgx**)
- Internal or external signals to synchronization inputs (**dmamux\_syncx**)

### 10.4.2 DMAMUX signals

*Table 37* lists the DMAMUX signals.

**Table 37. DMAMUX signals**

Signal name	Description
dmamux_hclk	DMAMUX AHB clock
dmamux_req_inx	DMAMUX DMA request line inputs from peripherals
dmamux_trgx	DMAMUX DMA request triggers inputs (to request generator sub-block)
dmamux_req_genx	DMAMUX request generator sub-block channels outputs
dmamux_reqx	DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)
dmamux_syncx	DMAMUX synchronization inputs (to request multiplexer sub-block)
dmamux_req_outx	DMAMUX requests outputs (to DMA controller)
dmamux_evtx	DMAMUX events outputs

### 10.4.3 DMAMUX channels

A DMAMUX channel is a request multiplexer channel that can include, depending upon the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to a single channel of DMA controller.

#### Channel configuration procedure

Follow the sequence below to configure a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

### 10.4.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals, or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ\_ID field in the DMAMUX\_CxCR register.

*Note:* *The null value in the field DMAREQ\_ID corresponds to no DMA request line selected.*

**Caution:** A same non-null DMAREQ\_ID cannot be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX\_CxCR and DMAMUX\_CyCR), except when the application guarantees that the two connected DMA channels are not simultaneously active.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

### Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX\_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC\_ID field in the DMAMUX\_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once a programmable rising/falling edge is detected on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX\_CxCR register.

Additionally, internally to the DMAMUX request multiplexer, there is a programmable DMA request counter, which can be used for the channel request output generation, and for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX\_CxCR register.

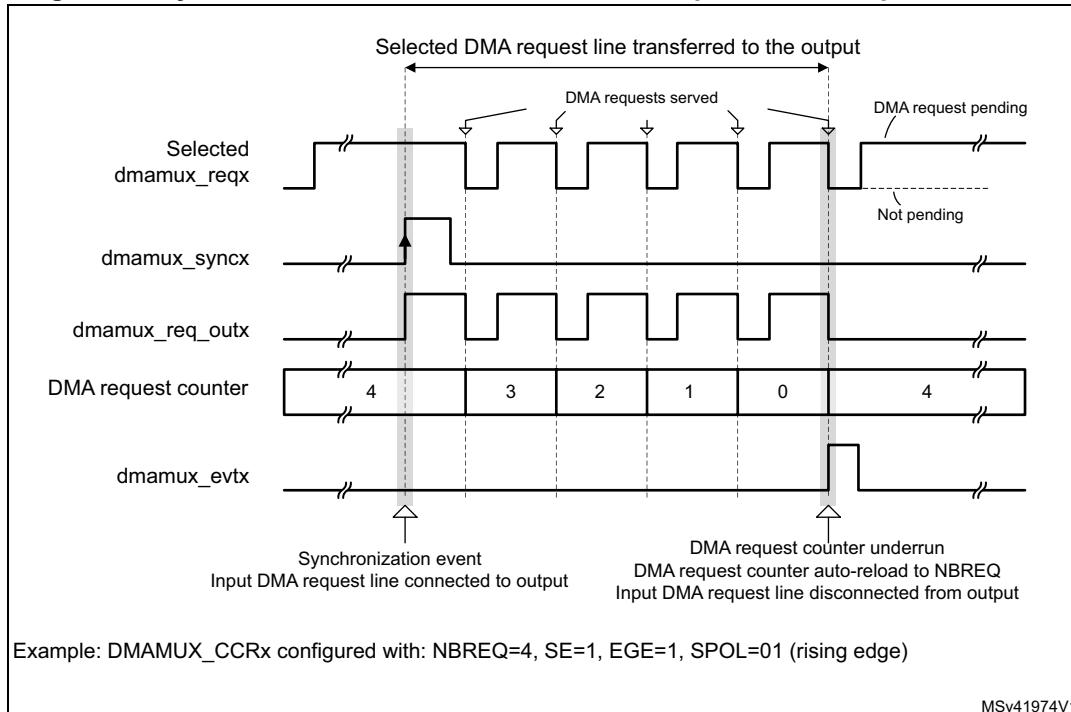
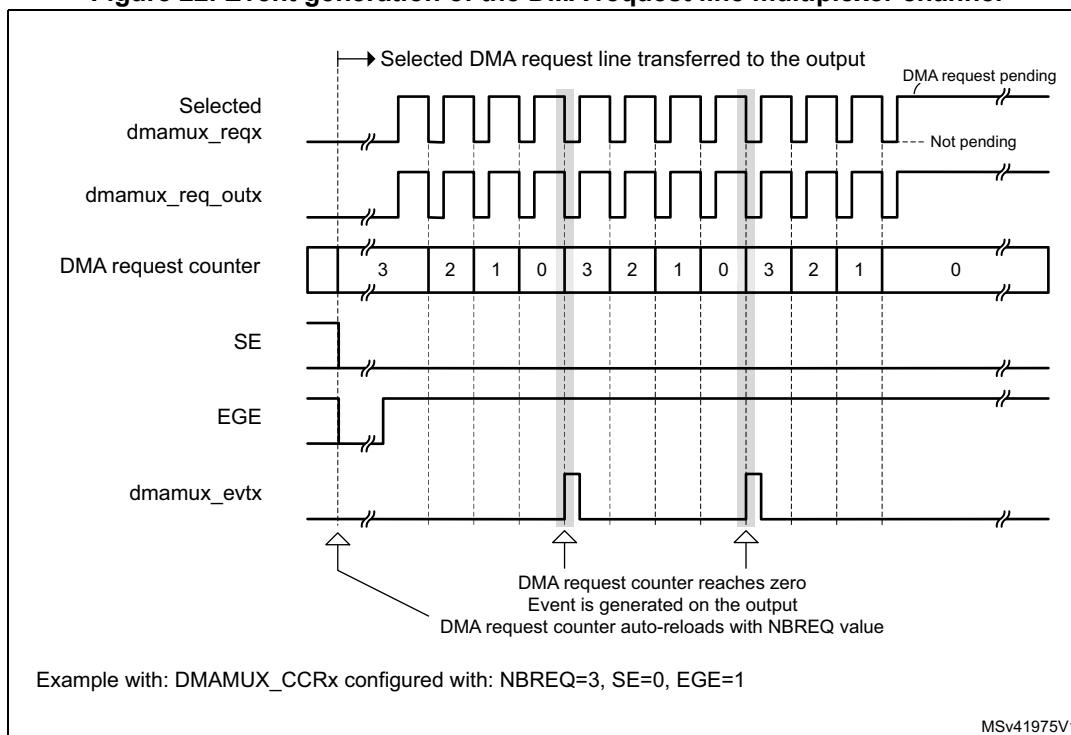
As shown in [Figure 22](#), upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

**Note:** *If a synchronization event occurs while there is no pending selected input DMA request line, it is discarded. The following asserted input request lines is not connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.*

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in the NBREQ field of the DMAMUX\_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in the NBREQ field, plus one.

**Note:** *The NBREQ field value can be written by software only when both synchronization enable bit (SE) and event generation enable bit (EGE) of the corresponding multiplexer channel x are disabled.*

**Figure 21. Synchronization mode of the DMAMUX request line multiplexer channel****Figure 22. Event generation of the DMA request line multiplexer channel**

If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in [Figure 21](#) and [Figure 22](#).

- Note:** If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.
- Note:** A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.
- Upon writing into DMAMUX\_CxCR register, the synchronization events are masked during three AHB clock cycles.

### Synchronization overrun and interrupt

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX\_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX\_CSR register.

- Note:** The request multiplexer channel x synchronization must be disabled (DMAMUX\_CxCR.SE = 0) when the use of the related channel of the DMA controller is completed. Else, upon a new detected synchronization event, there is a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX\_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX\_CxCR register.

## 10.4.5 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX\_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG\_ID (trigger signal ID) field in the corresponding DMAMUX\_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX\_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is deasserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter is automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.

**Note:** The GNBREQ field value can be written by software only when the enable GE bit of the corresponding generator channel x is disabled.

There is no hardware write protection.

A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX\_RGxCR register, the trigger events are masked during three AHB clock cycles.

### Trigger overrun and interrupt

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX\_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the DMAMUX\_RGSR register.

**Note:** The request generator channel x must be disabled (DMAMUX\_RGxCR.GE = 0) when the usage of the related channel of the DMA controller is completed. Else, upon a new detected trigger event, there is a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX\_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX\_RGxCR register.

## 10.5 DMAMUX interrupts

An interrupt can be generated upon:

- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel individual interrupt enable, status, and clear flag register bits are available.

Table 38. DMAMUX interrupts

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamuxovr_it	Synchronization event overrun on channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on channel x of the DMAMUX request generator	OFx	COFx	OIE

## 10.6 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX base address.

DMAMUX registers may be accessed per byte (8-bit), half-word (16-bit), or word (32-bit). The address must be aligned with the data size.

### 10.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX\_CxCR)

Address offset: 0x000 + 0x04 \* x (x = 0 to 2)

Reset value: 0x0000 0000

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SYNC\_ID[4:0]**: Synchronization identification

Selects the synchronization input (see [Table 36: DMAMUX: assignment of synchronization inputs to resources](#)).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward

Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.

This field must only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity

Defines the edge polarity of the selected synchronization input:

00: No event (no synchronization, no detection).

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 16 **SE**: Synchronization enable

0: Synchronization disabled

1: Synchronization enabled

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

Bit 9 **EGE**: Event generation enable

0: Event generation disabled

1: Event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable

0: Interrupt disabled

1: Interrupt enabled

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

Bits 5:0 **DMAREQ\_ID[5:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

### 10.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX\_CSR)

Address offset: 0x080

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.	SOF2	SOF1	SOF0												
													r	r	r

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

Bits 2:0 **SOF[2:0]**: Synchronization overrun event flag

The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.

The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX\_CFR register.

### 10.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX\_CFR)

Address offset: 0x084

Reset value: 0x0000 0000

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.	CSOF2	CSOF1	CSOF0												
													w	w	w

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

Bits 2:0 **CSOF[2:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX\_CSR register.

### 10.6.4 DMAMUX request generator channel x configuration register (DMAMUX\_RGxCR)

Address offset: 0x100 + 0x04 \* x (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	GNBREQ[4:0]				GPOL[1:0]		GE								
								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.	OIE	Res.	Res.	Res.	Res.	SIG_ID[4:0]									
							rw					rw	rw	rw	rw

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

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

*Note: This field must be written only when GE bit is disabled.*

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input

00: No event, i.e. no trigger detection nor generation.

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled

1: DMA request generator channel x enabled

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

Bit 8 **OIE**: Trigger overrun interrupt enable

0: Interrupt on a trigger overrun event occurrence is disabled

1: Interrupt on a trigger overrun event occurrence is enabled

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

Bits 4:0 **SIG\_ID[4:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

### 10.6.5 DMAMUX request generator interrupt status register (DMAMUX\_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

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.	OF3	OF2	OF1	OF0											
												r	r	r	r

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

Bits 3:0 **OF[3:0]**: Trigger overrun event flag

The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX\_RGxCR register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX\_RGCFR register.

### 10.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX\_RGCFR)

Address offset: 0x144

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.	w	w	w	w											

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

Bits 3:0 **COF[3:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX\_RGSR register.

## 10.6.7 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

**Table 39. DMAMUX register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x000	DMAMUX_C0CR	Res.	Res.	Res.	Sync_ID[4:0]		NBREQ[4:0]	SPOL[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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0				
0x004	DMAMUX_C1CR	Res.	Res.	Res.	Sync_ID[4:0]	NBREQ[4:0]	SPOL[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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0				
0x008	DMAMUX_C2CR	Res.	Res.	Res.	Sync_ID[4:0]	NBREQ[4:0]	SPOL[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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x0 - 0x07C	Reserved	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.				
0x080	DMAMUX_CSR																																	
	Reset value	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x084	DMAMUX_CFR																																	
	Reset value	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x088 - 0x0FC	Reserved																																	
0x100	DMAMUX_RG0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]	GPOL[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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x104	DMAMUX_RG1CR																	GNBREQ[4:0]	GPOL[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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x108	DMAMUX_RG2CR																	GNBREQ[4:0]	GPOL[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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x10C	DMAMUX_RG3CR																	GNBREQ[4:0]	GPOL[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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0				
0x110 - 0x13C	Reserved																	OIE	0	OIE	0	OIE	0	OIE	0	OIE	0	OIE	0	OIE	0	OIE	0	OIE
0x140	DMAMUX_RGSR	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 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x144	DMAMUX_RGCFR																																	
	Reset value	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0			
0x148 - 0x3FC	Reserved	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.			

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 11 Nested vectored interrupt controller (NVIC)

### 11.1 Main features

- 32 maskable interrupt channels (not including the sixteen Cortex®-M0+ interrupt lines)
- 4 programmable priority levels (2 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, which enables low-latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the programming manual PM0223.

### 11.2 SysTick calibration value register

The SysTick calibration value is set to 1000. SysTick reload value register may be adapted to the actual HCLK frequency and required time period, see PM0223 for more details.

### 11.3 Interrupt and exception vectors

*Table 40* is the vector table. Information pertaining to a peripheral only applies to devices containing that peripheral.

**Table 40. Vector table<sup>(1)</sup>**

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000_0000
-	-3	fixed	Reset	Reset	0x0000_0004
-	-2	fixed	NMI_Handler	Non maskable interrupt. SRAM parity error, HSE CSS and LSE CSS are linked to the NMI vector.	0x0000_0008
-	-1	fixed	HardFault_Handler	All class of fault	0x0000_000C
-	-	-	-	Reserved	0x0000_0010 0x0000_0014 0x0000_0018 0x0000_001C 0x0000_0020 0x0000_0024 0x0000_0028
-	3	settable	SVC_Handler	System service call via SWI instruction	0x0000_002C

**Table 40. Vector table<sup>(1)</sup> (continued)**

<b>Position</b>	<b>Priority</b>	<b>Type of priority</b>	<b>Acronym</b>	<b>Description</b>	<b>Address</b>
-	-	-	-	Reserved	0x0000_0030 0x0000_0034
-	5	settable	PendSV_Handler	Pendable request for system service	0x0000_0038
-	6	settable	SysTick_Handler	System tick timer	0x0000_003C
0	7	settable	WWDG	Window watchdog interrupt	0x0000_0040
1	-	-	-	Reserved	0x0000_0044
2	9	settable	RTC	RTC interrupts (EXTI lines 19)	0x0000_0048
3	10	settable	FLASH	Flash global interrupt	0x0000_004C
4	11	settable	RCC	RCC global interrupt	0x0000_0050
5	12	settable	EXTI0_1	EXTI line 0 & 1 interrupt	0x0000_0054
6	13	settable	EXTI2_3	EXTI line 2 & 3 interrupt	0x0000_0058
7	14	settable	EXTI4_15	EXTI line 4 to 15 interrupt	0x0000_005C
8	-	-	-	Reserved	0x0000_0060
9	16	settable	DMA1_Channel1	DMA1 channel 1 interrupt	0x0000_0064
10	17	settable	DMA1_Channel2_3	DMA1 channel 2 & 3 interrupts	0x0000_0068
11	18	settable	DMAMUX	DMAMUX interrupts	0x0000_006C
12	19	settable	ADC	ADC interrupt	0x0000_0070
13	20	settable	TIM1_BRK_UP_TRG _COM	TIM1 break, update, trigger and commutation interrupts	0x0000_0074
14	21	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_0078
15	-	-	-	Reserved	0x0000_007C
16	23	settable	TIM3	TIM3 global interrupt	0x0000_0080
17	-	-	-	Reserved	0x0000_0084
18	-	-	-	Reserved	0x0000_0088
19	26	settable	TIM14	TIM14 global interrupt	0x0000_008C
20	-	-	-	Reserved	0x0000_0090
21	28	settable	TIM16	TIM16 global interrupt	0x0000_0094
22	29	settable	TIM17	TIM17 global interrupt	0x0000_0098
23	30	settable	I2C1	I2C1 global interrupt (combined with EXTI 23)	0x0000_009C
24	-	-	-	Reserved	0x0000_00A0
25	32	settable	SPI1	SPI1/I2S1 global interrupt	0x0000_00A4
26	-	-	-	Reserved	0x0000_00A8
27	34	settable	USART1	USART1 global interrupt (combined with EXTI 25)	0x0000_00AC

**Table 40. Vector table<sup>(1)</sup> (continued)**

<b>Position</b>	<b>Priority</b>	<b>Type of priority</b>	<b>Acronym</b>	<b>Description</b>	<b>Address</b>
28	35	settable	USART2	USART2 global interrupt	0x0000_00B0
29	-	-	-	Reserved	0x0000_00B4
30	-	-	-	Reserved	0x0000_00B8
31	-	-	-	Reserved	0x0000_00BC

1. The grayed cells correspond to the Cortex®-M0+ interrupts.

## 12

# Extended interrupt and event controller (EXTI)

The Extended interrupt and event controller (EXTI) manages the CPU and system wakeup through configurable and direct event inputs (lines). It provides wakeup requests to the power control, and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can also be used in Run mode.

The EXTI also includes the EXTI I/O port multiplexer.

## 12.1

### EXTI main features

The EXTI main features are the following:

- System wakeup upon event on any input
- Wakeup flag and CPU interrupt generation for events not having a wakeup flag in their source peripheral
- Configurable events (from I/Os, peripherals not having an associated interrupt pending status bit, or peripherals generating a pulse)
  - Selectable active trigger edge
  - Independent rising and falling edge interrupt pending status bits
  - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
  - SW trigger possibility
- Direct events (from peripherals having an associated flag and interrupt pending status bit)
  - Fixed rising edge active trigger
  - No interrupt pending status bit in the EXTI
  - Individual interrupt and event generation mask for conditioning the CPU wakeup and event generation
  - No SW trigger possibility
- I/O port selector

## 12.2

### EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block, and EXTI multiplexer as shown in [Figure 23](#).

The register block contains all the EXTI registers.

The event input trigger block provides an event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and the masking of these.

The EXTI multiplexer provides the I/O port selection on to the EXTI event signal.

Figure 23. EXTI block diagram

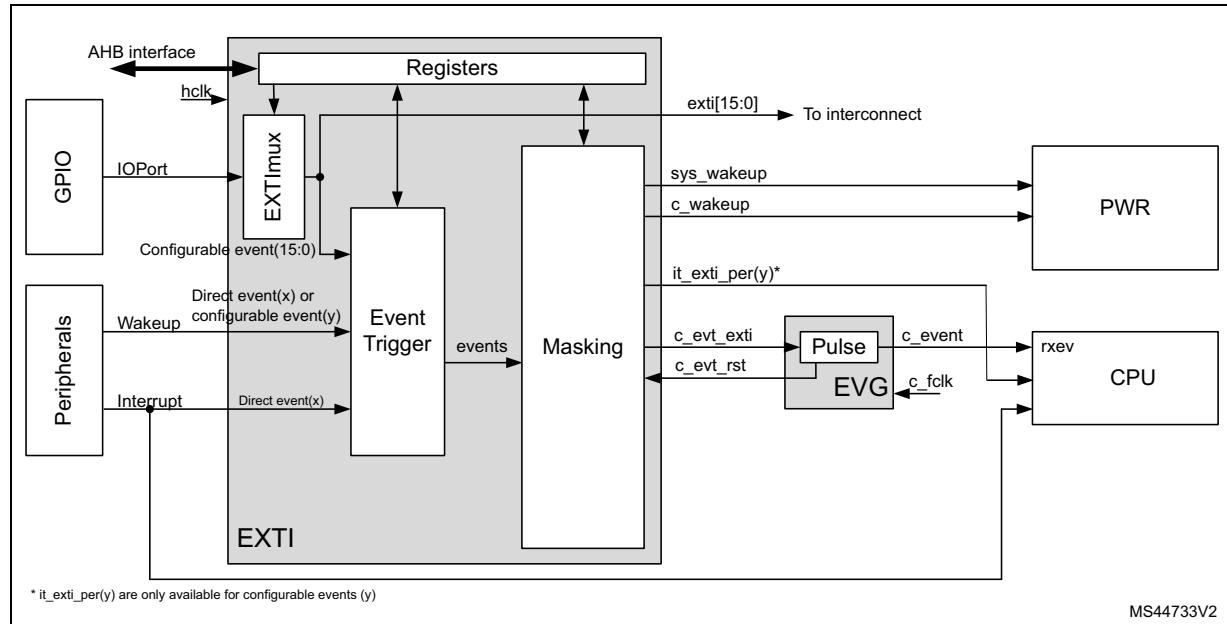


Table 41. EXTI signal overview

Signal name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to allow security, the AHB interface support secure accesses
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wakeup events from peripherals that do not have an associated interrupt and flag in the peripheral
Direct event(x)	I	Synchronous and asynchronous wakeup events from peripherals having an associated interrupt and flag in the peripheral
IOPort(n)	I	GPIO ports[15:0]
exti[15:0]	O	EXTI output port to trigger other IPs
it_exti_per (y)	O	Interrupts to the CPU associated with configurable event (y)
c_evt_exti	O	High-level sensitive event output for CPU synchronous to hclk
c_evt_RST	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wakeup request to PWR for ck_sys and hclk
c_wakeup	O	Wakeup request to PWR for CPU, synchronous to hclk

Table 42. EVG pin overview

Pin name	I/O	Description
c_fclk	I	CPU free-running clock
c_evt_in	I	High-level sensitive event input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_RST	O	Event reset signal, synchronous to CPU clock

### 12.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in Stop mode are connected to the EXTI.

- Peripheral wakeup signals that generate a pulse or that do not have an interrupt status bits in the peripheral, are connect to an EXTI configurable line. For these events the EXTI provides a status pending bit which requires to be cleared. It is the EXTI interrupt associated with the status bit that interrupts the CPU.
- Peripheral interrupt and wakeup signals that have a status bit in the peripheral which requires to be cleared in the peripheral, are connected to an EXTI direct line. There is no status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.
- All GPIO ports input to the EXTI multiplexer, allowing to select a port to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC(a) of the CPU.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wakeup signals are connected to the PWR block, and are used to wake up the system and CPU sub-system bus clocks.

### 12.3 EXTI functional description

Depending on the EXTI line type and wakeup target(s), different logic implementations are used. The applicable features and control or status registers are:

- rising and falling edge event enable through
  - *EXTI rising trigger selection register (EXTI\_RTSR1)*
  - *EXTI falling trigger selection register 1 (EXTI\_FTSR1)*
- software trigger through *EXTI software interrupt event register 1 (EXTI\_SWIER1)*
- pending interrupt flagging through
  - *EXTI rising edge pending register 1 (EXTI\_RPR1)*
  - *EXTI falling edge pending register 1 (EXTI\_FPR1)*
  - *EXTI external interrupt selection register (EXTI\_EXTICRx)*
- CPU wakeup and interrupt enable through
  - *EXTI CPU wakeup with interrupt mask register (EXTI\_IMR1)*
- CPU wakeup and event enable through
  - *EXTI CPU wakeup with event mask register (EXTI\_EMR1)*

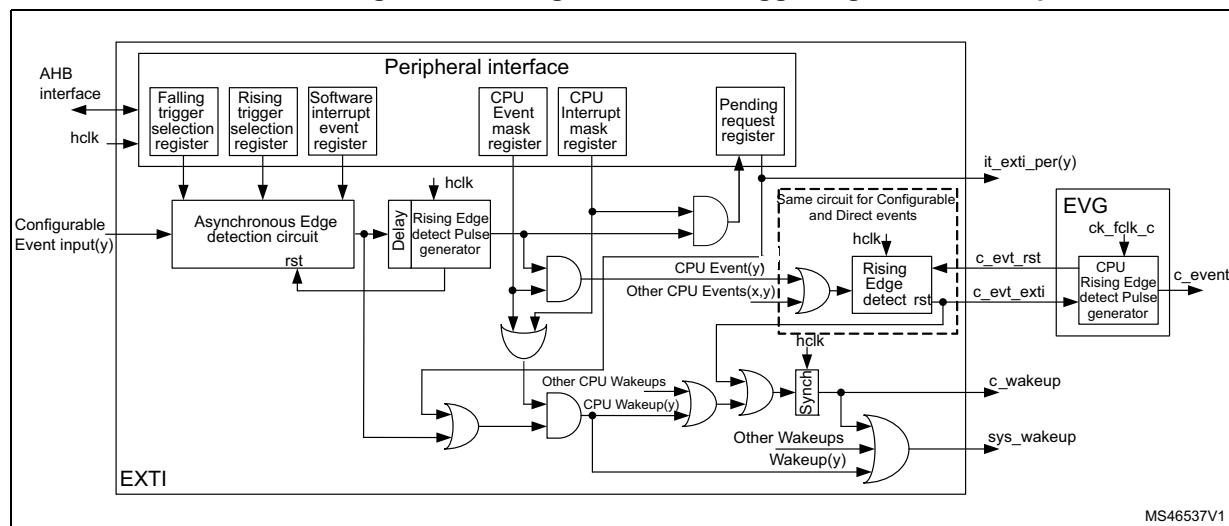
**Table 43. EXTI event input configurations and register control**

Event input type	Logic implementation	EXTI_RTSR1	EXTI_FTSR1	EXTI_SWIER1	EXTI_RPR1	EXTI_IMR1	EXTI_EMR1
Configurable	Configurable event input wakeup logic	x	x	x	x	x	x
Direct	Direct event input wakeup logic	-	-	-	-	x	x

### 12.3.1 EXTI configurable event input wakeup

*Figure 24* is a detailed representation of the logic associated with configurable event inputs which wake up the CPU sub-system bus clocks and generated an EXTI pending flag and interrupt to the CPU and or a CPU wakeup event.

**Figure 24. Configurable event trigger logic CPU wakeup**



The software interrupt event register allows triggering configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising edge and falling edge selection registers allow to enable and select the configurable event active trigger edge or both edges.

The CPU has its dedicated interrupt mask register and a dedicated event mask registers. The enabled event allows generating an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The event pending registers (EXTI\_RPR1 and EXTI\_FPR1) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPU. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts need to be acknowledged by software in the EXTI\_RPR1 and/or EXTI\_FPR1 registers.

When a CPU interrupt or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees the wakeup of the EXTI hclk clock before the asynchronous edge detection circuit is reset.

**Note:** A detected configurable event interrupt pending request can be cleared by the CPU. The system cannot enter low-power modes as long as an interrupt pending request is active.

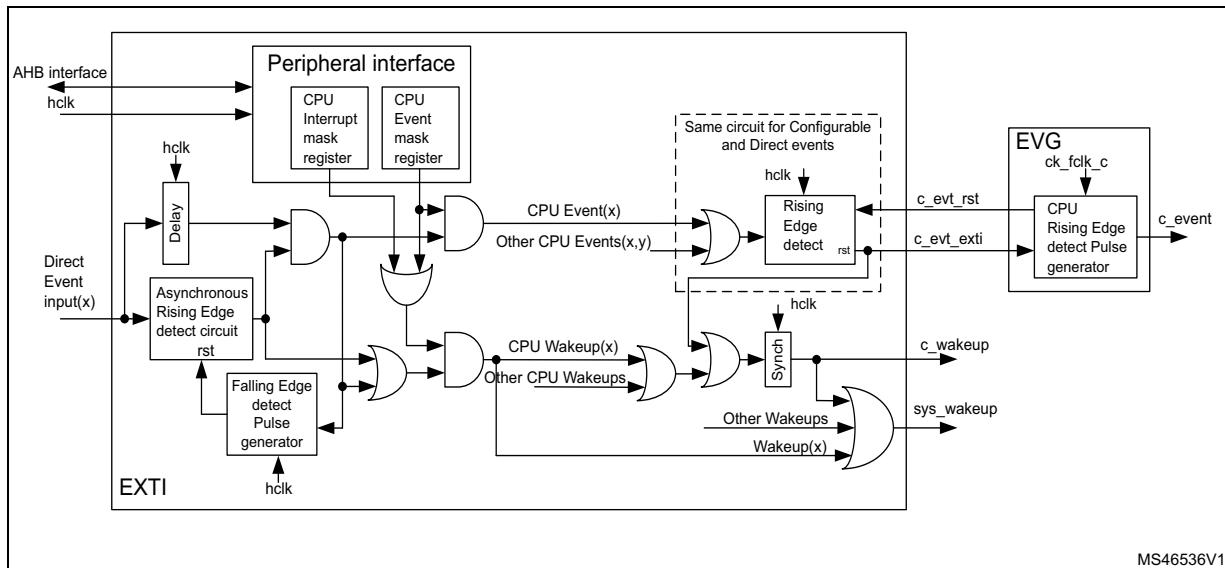
### 12.3.2 EXTI direct event input wakeup

*Figure 25* is a detailed representation of the logic associated with direct event inputs waking up the system.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU sub-system clocks and may generate a CPU wakeup event. The peripheral synchronous interrupt, associated with the direct wakeup event wakes up the CPU.

The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU. The CPU event may occur before the interrupt flag of the associated peripheral is set.

**Figure 25. Direct event trigger logic CPU wakeup**

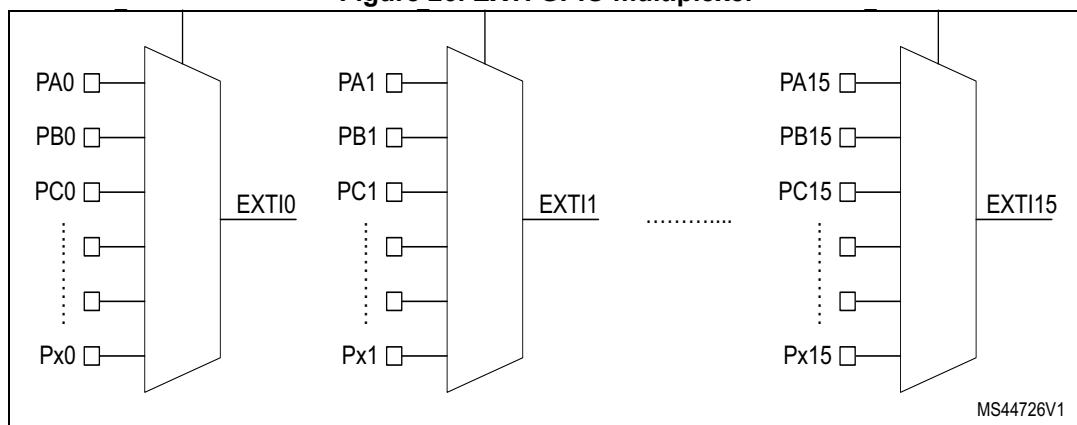


MS46536V1

### 12.3.3 EXTI multiplexer

The EXTI multiplexer allows selecting GPIOs as interrupts and wakeup. The GPIOs are connected via 16 EXTI multiplexer lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI multiplexer output is controlled through the [EXTI external interrupt selection register \(EXTI\\_EXTICR<sub>x</sub>\)](#) register.

**Figure 26. EXTI GPIO multiplexer**



MS44726V1

The EXTIs multiplexer outputs are available as output signals from the EXTI, to trigger other functional blocks. The EXTI multiplexer outputs are available independently of mask setting through the EXTI\_IMR and EXTI\_EMR registers.

The EXTI lines (event inputs) are connected as shown in the following table.

**Table 44. EXTI line connections**

EXTI line	Line source	Line type
0-15	GPIO	Configurable
16	Reserved	-
17	Reserved	-
18	Reserved	-
19	RTC	Direct
20	Reserved	-
21	Reserved	-
22	Reserved	-
23	I2C1 wakeup	Direct
24	Reserved	-
25	USART1 wakeup	Direct
26	Reserved	-
27	Reserved	-
28	Reserved	-
29	Reserved	-
30	Reserved	-
31	LSE_CSS	Direct
32	Reserved	-
33	Reserved	-

## 12.4 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wakeup event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of a CPU wakeup is conditioned by the CPU interrupt mask and CPU event mask.

**Table 45. Masking functionality**

CPU interrupt enable EXTI_IMR.IMn	CPU event enable EXTI_EMR.EMn	Configurable event inputs EXTI_RPR.RPIFn EXTI_FPR.FPIFn	exti(n) interrupt <sup>(1)</sup>	CPU event	CPU wakeup
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes <sup>(2)</sup>
	1	Status latched	Yes	Yes	Yes

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU, the exti(n) interrupt must be masked in the CPU NVIC.

2. Only if CPU interrupt is enabled in EXTI\_IMR.IMn.

For configurable event inputs, upon an edge on the event input, an event request is generated if that edge (rising or/and falling) is enabled. When the associated CPU interrupt is unmasked, the corresponding RPIFn and/or FPIFn bit is/are set in the EXTI\_RPR or/and EXTI\_FPR register, waking up the CPU subsystem and activating CPU interrupt signal. The RPIFn and/or FPIFn pending bit is cleared by writing 1 to it, which clears the CPU interrupt request.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI. When the associated CPU interrupt is unmasked, the corresponding CPU subsystem is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU event must be unmasked to generate an event. Upon an enabled edge occurring on an event input, a CPU event pulse is generated. There is no event pending bit.

For the configurable event inputs, the software can generate an event request by setting the corresponding bit of the software interrupt/event register EXTI\_SWIER1, which has the effect of a rising edge on the event input. The pending rising edge event flag is set in the EXTI\_RPR1 register, irrespective of the EXTI\_RTSR1 register setting.

## 12.5 EXTI registers

The EXTI register map is divided in the following sections:

**Table 46. EXTI register map sections**

Address	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x060 - 0x06C	EXTI I/O port multiplexer
0x080 - 0x0BC	CPU input event configuration

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

### 12.5.1 EXTI rising trigger selection register (EXTI\_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

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
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw															

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

Bits 15:0 **RTx**: Rising trigger event configuration bit of configurable line x (x = 15 to 0)<sup>(1)</sup>

Each bit enables/disables the rising edge trigger for the event and interrupt on the corresponding line.

0: Disable

1: Enable

- The configurable lines are edge triggered, no glitch must be generated on these inputs.

If a rising edge on the configurable line occurs during writing of the register, the associated pending bit is not set. Rising edge trigger can be set for a line with falling edge trigger enabled. In this case, both edges generate a trigger.

## 12.5.2 EXTI falling trigger selection register 1 (EXTI\_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

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
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw															

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

Bits 15:0 **FTx**: Falling trigger event configuration bit of configurable line x (x = 15 to 0)<sup>(1)</sup>.

Each bit enables/disables the falling edge trigger for the event and interrupt on the corresponding line.

0: Disable

1: Enable

- The configurable lines are edge triggered, no glitch must be generated on these inputs.

If a falling edge on the configurable line occurs during writing of the register, the associated pending bit is not set. Falling edge trigger can be set for a line with rising edge trigger enabled. In this case, both edges generate a trigger.

## 12.5.3 EXTI software interrupt event register 1 (EXTI\_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

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
SWI 15	SWI 14	SWI 13	SWI 12	SWI 11	SWI 10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
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 **SWIx**: Software rising edge event trigger on line x (x = 15 to 0)

Setting of any bit by software triggers a rising edge event on the corresponding line x, resulting in an interrupt, independently of EXTI\_RTSR1 and EXTI\_FTSR1 settings. The bits are automatically cleared by HW. Reading of any bit always returns 0.

0: No effect

1: Rising edge event generated on the corresponding line, followed by an interrupt

#### 12.5.4 EXTI rising edge pending register 1 (EXTI\_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

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
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 15:0 **RPIFx**: Rising edge event pending for configurable line x (x = 15 to 0)

Each bit is set upon a rising edge event generated by hardware or by software (through the EXTI\_SWIER1 register) on the corresponding line. Each bit is cleared by writing 1 into it.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

#### 12.5.5 EXTI falling edge pending register 1 (EXTI\_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

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
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 15:0 **FPIFx**: Falling edge event pending for configurable line x ( $x = 15$  to 0)

Each bit is set upon a falling edge event generated by hardware or by software (through the EXTI\_SWIER1 register) on the corresponding line. Each bit is cleared by writing 1 into it.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

## 12.5.6 EXTI external interrupt selection register (EXTI\_EXTICRx)

Address offset:  $0x060 + 0x4 * (x - 1)$ , ( $x = 1$  to 4)

Reset value: 0x0000 0000

EXTIm fields contain only the number of bits in line with the nb\_iport configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTIm+3[7:0]								EXTIm+2[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
EXTIm+1[7:0]								EXTIm[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTIm+3[7:0]**: EXTIm+3 GPIO port selection ( $m = 4 * (x - 1)$ )

These bits are written by software to select the source input for EXTIm+3 external interrupt.

0x00: PA[m+3] pin

0x01: PB[m+3] pin

0x02: PC[m+3] pin

0x03: PD[m+3] pin

0x04: reserved

0x05: PF[m+3] pin

Other: reserved

Bits 23:16 **EXTIm+2[7:0]**: EXTIm+2 GPIO port selection ( $m = 4 * (x - 1)$ )

These bits are written by software to select the source input for EXTIm+2 external interrupt.

0x00: PA[m+2] pin

0x01: PB[m+2] pin

0x02: PC[m+2] pin

0x03: PD[m+2] pin

0x04: reserved

0x05: PF[m+2] pin

Other: reserved

Bits 15:8 **EXTIm+1[7:0]**: EXTIm+1 GPIO port selection ( $m = 4 * (x - 1)$ )

These bits are written by software to select the source input for EXTIm+1 external interrupt.

0x00: PA[m+1] pin

0x01: PB[m+1] pin

0x02: PC[m+1] pin

0x03: PD[m+1] pin

0x04: reserved

0x05: PF[m+1] pin

Other: reserved

Bits 7:0 **EXTIm[7:0]**: EXTIm GPIO port selection ( $m = 4 * (x - 1)$ )

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

- 0x00: PA[m] pin
- 0x01: PB[m] pin
- 0x02: PC[m] pin
- 0x03: PD[m] pin
- 0x04: reserved
- 0x05: PF[m] pin
- Other: reserved

### 12.5.7 EXTI CPU wakeup with interrupt mask register (EXTI\_IMR1)

Address offset: 0x080 (EXTI\_IMR1)

Reset value: 0xFFFF8 0000

Contains register bits for configurable events and direct events.

The reset value is set such as to, by default, enable interrupt from direct lines, and disable interrupt from configurable lines.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	Res.	Res.	Res.	Res.	Res.	IM25	Res.	IM23	Res.	Res.	Res.	IM19	Res.	Res.	Res.
rw						rw		rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw															

Bit 31 **IM31**: CPU wakeup with interrupt mask on line 31

Setting/clearing this bit unmasks/masks the CPU wakeup with interrupt, by an event on the corresponding line.

- 0: wakeup with interrupt masked
- 1: wakeup with interrupt unmasked

Bits 30:26 Reserved, must be kept at reset value.

Bit 25 **IM25**: CPU wakeup with interrupt mask on line 25

Setting/clearing each bit unmasks/masks the CPU wakeup with interrupt, by an event on the corresponding line.

- 0: wakeup with interrupt masked
- 1: wakeup with interrupt unmasked

Bit 24 Reserved, must be kept at reset value.

Bit 23 **IM23**: CPU wakeup with interrupt mask on line 23

Setting/clearing each bit unmasks/masks the CPU wakeup with interrupt, by an event on the corresponding line.

- 0: wakeup with interrupt masked
- 1: wakeup with interrupt unmasked

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **IM19:** CPU wakeup with interrupt mask on line 19

Setting/clearing this bit unmasks/masks the CPU wakeup with interrupt, by an event on the corresponding line.

0: wakeup with interrupt masked

1: wakeup with interrupt unmasked

Bits 18:16 Reserved, must be kept at reset value.

Bits 15:0 **IMx:** CPU wakeup with interrupt mask on line x (x = 15 to 0)

Setting/clearing each bit unmasks/masks the CPU wakeup with interrupt, by an event on the corresponding line.

0: wakeup with interrupt masked

1: wakeup with interrupt unmasked

### 12.5.8 EXTI CPU wakeup with event mask register (EXTI\_EMR1)

Address offset: 0x084 (EXTI\_EMR1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	Res.	Res.	Res.	Res.	Res.	EM25	Res.	EM23	Res.	Res.	Res.	EM19	Res.	Res.	Res.
rw						rw		rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw															

Bit 31 **EM31:** CPU wakeup with event generation mask on line 31

Setting/clearing this bit unmasks/masks the CPU wakeup with event generation on the corresponding line.

0: wakeup with event generation masked

1: wakeup with event generation unmasked

Bits 30:26 Reserved, must be kept at reset value.

Bit 25 **EM25:** CPU wakeup with event generation mask on line 25

Setting/clearing this bit unmasks/masks the CPU wakeup with event generation on the corresponding line.

0: wakeup with event generation masked

1: wakeup with event generation unmasked

Bit 24 Reserved, must be kept at reset value.

Bit 23 **EM23:** CPU wakeup with event generation mask on line 23

Setting/clearing this bit unmasks/masks the CPU wakeup with event generation on the corresponding line.

0: wakeup with event generation masked

1: wakeup with event generation unmasked

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 **EM19:** CPU wakeup with event generation mask on line 19

Setting/clearing this bit unmasks/masks the CPU wakeup with event generation on the corresponding line.

0: wakeup with event generation masked

1: wakeup with event generation unmasked

Bits 18:16 Reserved, must be kept at reset value.

Bits 15:0 **EMx:** CPU wakeup with event generation mask on line x (x = 15 to 0)

Setting/clearing each bit unmasks/masks the CPU wakeup with event generation on the corresponding line.

0: wakeup with event generation masked

1: wakeup with event generation unmasked

## 12.5.9 EXTI register map

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

**Table 47. EXTI 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
0x000	<b>EXTI_RTSR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.													
	Reset value																																
0x004	<b>EXTI_FTSR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.														
	Reset value																																
0x008	<b>EXTI_SWIER1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.														
	Reset value																																
0x00C	<b>EXTI_RPR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.														
	Reset value																																
0x010	<b>EXTI_FPR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.														
	Reset value																																
0x014-0x05C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.														
0x060	<b>EXTI_EXTICR1</b>	EXTI3[7:0]				EXTI2[7:0]				EXTI1[7:0]				EXTI0[7:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x064	<b>EXTI_EXTICR2</b>	EXTI17[7:0]				EXTI16[7:0]				EXTI15[7:0]				EXTI14[7:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x068	<b>EXTI_EXTICR3</b>	EXTI11[7:0]				EXTI10[7:0]				EXTI9[7:0]				EXTI8[7:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x06C	<b>EXTI_EXTICR4</b>	EXTI15[7:0]				EXTI14[7:0]				EXTI13[7:0]				EXTI12[7:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**Table 47. EXTI controller register map and reset values (continued)**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x080	<b>EXTI_IMR1</b>	IM31	Res.	IM23	Res.	Res.	Res.	IM19	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
	Reset value	1	Res.	1	Res.	Res.	Res.	0	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x084	<b>EXTI_EMR1</b>	EM31	Res.	EM23	Res.	Res.	Res.	EM19	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
	Reset value	0	Res.	0	Res.	Res.	Res.	0	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x088-0x08C	Reserved	Res.																																

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 13 Cyclic redundancy check calculation unit (CRC)

### 13.1 Introduction

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

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

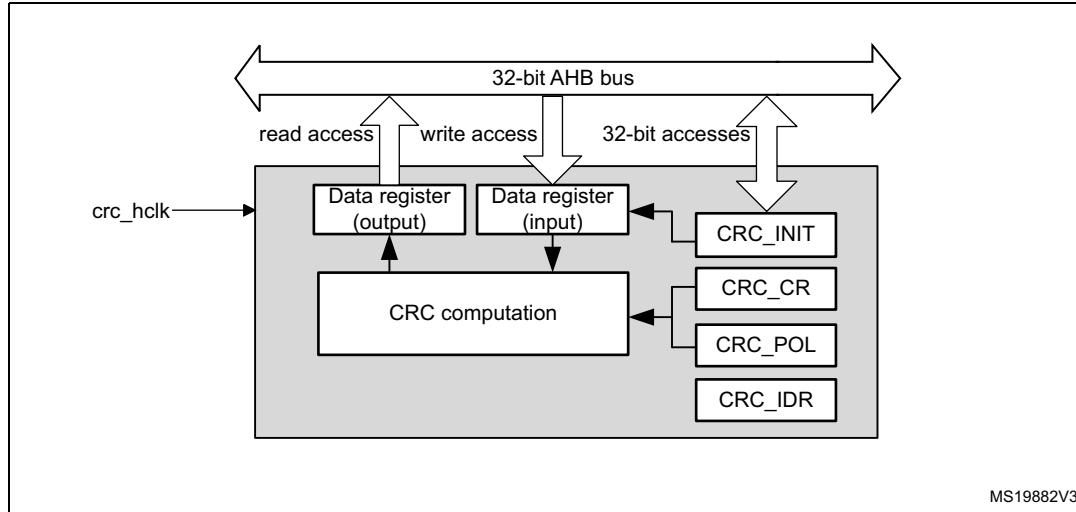
### 13.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-, 16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC\_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

## 13.3 CRC functional description

### 13.3.1 CRC block diagram

Figure 27. CRC calculation unit block diagram



### 13.3.2 CRC internal signals

Table 48. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

### 13.3.3 CRC operation

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

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

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

The duration of the computation depends on data width:

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

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

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

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

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

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

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

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

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

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

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

## Polynomial programmability

The polynomial coefficients are fully programmable through the CRC\_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC\_CR register. Even polynomials are not supported.

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

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

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size 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.

## 13.4 CRC registers

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

### 13.4.1 CRC data register (CRC\_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **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.

### 13.4.2 CRC 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
IDR[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
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC\_CR register

### 13.4.3 CRC 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.	Res.	RESET	rs							
								rw	rw	rw	rw	rw			

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

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

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

0: Bit order not affected

1: Bit-reversed output format

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

This bitfield controls the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

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

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

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

### 13.4.4 CRC initial value (CRC\_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

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

This register is used to write the CRC initial value.

### 13.4.5 CRC polynomial (CRC\_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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

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

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

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

### 13.4.6 CRC register map

Table 49. CRC 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	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		
0x04	CRC_IDR	IDR[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		
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.	RESET				
	Reset value																														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		
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	1	1		

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 14 Analog-to-digital converter (ADC)

### 14.1 Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 23 multiplexed channels allowing it to measure signals from 19 external and 4 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

A built-in hardware oversampler allows analog performances to be improved while off-loading the related computational burden from the CPU.

## 14.2 ADC main features

- High performance
  - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
  - ADC conversion time: 0.4  $\mu$ s for 12-bit resolution (2.5Msps), faster conversion times can be obtained by lowering resolution.
  - Self-calibration
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Low-power
  - The application can reduce PCLK frequency for low-power operation while still keeping optimum ADC performance. For example, 0.4  $\mu$ s conversion time is kept, whatever the PCLK frequency
  - Wait mode: prevents ADC overrun in applications with low PCLK frequency
  - Auto off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Analog input channels
  - Up to 19 external analog inputs
  - 1 channel for internal temperature sensor ( $V_{SENSE}$ )
  - 1 channel for internal reference voltage ( $V_{REFINT}$ )
  - 2 channels for monitoring internal power supply ( $V_{DDA}/V_{SSA}$ )
- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (timer events or GPIO input events)
- Conversion modes
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events
- Analog watchdog
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8-bits
- ADC input range:  $V_{SSA} \leq V_{IN} \leq V_{DDA}$

## 14.3 ADC implementation

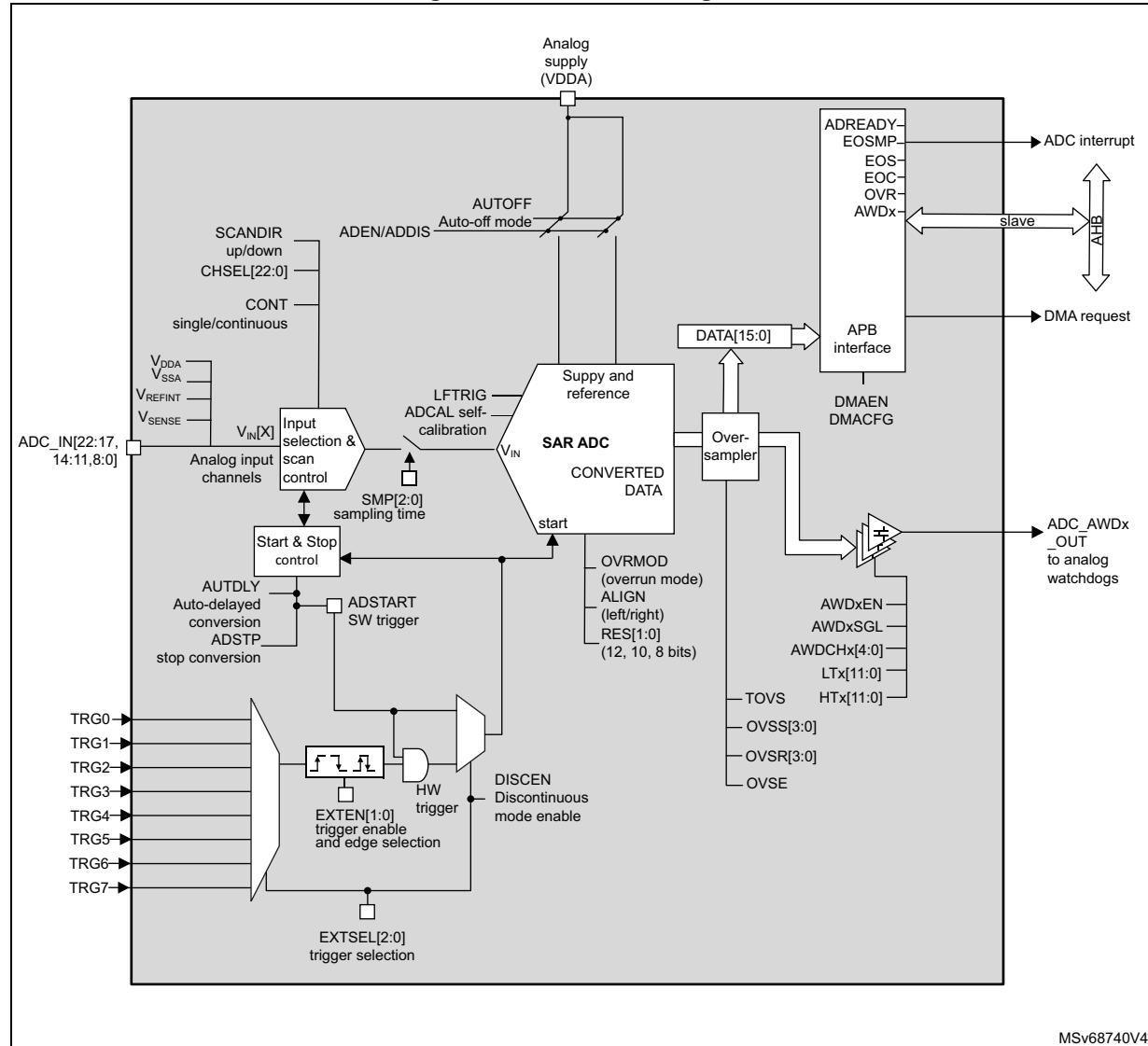
Table 50. ADC main features

ADC modes/features	STM32C01xx	STM32C03xx
Resolution	12 bits	
Maximum sampling speed	2.5 Msps	
Hardware offset calibration	X	
Single-ended inputs	X	
Differential inputs	-	
Oversampling mode	X	
Data register	16 bits	
DMA support	X	
Number of analog watchdogs	3	
Number of external channels	13	19
Number of internal channels		4

## 14.4 ADC functional description

Figure 28 shows the ADC block diagram and Table 51 gives the ADC pin description.

Figure 28. ADC block diagram



MSv68740V4

1. TRGi are mapped at product level. Refer to [Table External triggers](#) in Section 14.4.1: ADC pins and internal signals.

### 14.4.1 ADC pins and internal signals

Table 51. ADC input/output pins

Name	Signal type	Remarks
VDDA	Input, analog power supply	Analog power supply and positive reference voltage for the ADC

**Table 51. ADC input/output pins**

Name	Signal type	Remarks
VSSA	Input, analog supply ground	Ground for analog power supply
ADC_INx	Analog input signals	Up to 19 external analog input channels

**Table 52. ADC internal input/output signals**

Internal signal name	Signal type	Description
$V_{IN[x]}$	Analog Input channels	Connected either to internal channels or to ADC_IN/external channels
TRGx	Input	ADC conversion triggers
$V_{SENSE}$	Input	Internal temperature sensor output voltage
$V_{REFINT}$	Input	Internal voltage reference output voltage
ADC_AWDx_OUT	Output	Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number = 1,2,3)

**Table 53. External triggers**

Name	Source	EXTSEL[2:0]
TRG0	TIM1_TRGO2	000
TRG1	TIM1_CC4	001
TRG2	Reserved	010
TRG3	TIM3_TRGO	011
TRG4	Reserved	100
TRG5	Reserved	101
TRG6	Reserved	110
TRG7	EXTI11	111

#### 14.4.2 ADC voltage regulator (ADVREGEN)

The ADC has a specific internal voltage regulator which must be enabled and stable before using the ADC.

The ADC internal voltage regulator can be enabled by setting ADVREGEN bit to 1 in the ADC\_CR register. The software must wait for the ADC voltage regulator startup time ( $t_{ADCVREG\_STUP}$ ) before launching a calibration or enabling the ADC. This delay must be managed by software (for details on  $t_{ADCVREG\_STUP}$ , refer to the device datasheet).

After ADC operations are complete, the ADC is disabled (ADEN = 0). To keep power consumption low, it is important to disable the ADC voltage regulator before entering low-power mode (LPRun, LPSleep or Stop mode). Refer to [Section : ADC voltage regulator disable sequence](#).

*Note:* When the internal voltage regulator is disabled, the internal analog calibration is kept.

### Analog reference from the power control unit

The internal ADC voltage regulator internally uses an analog reference delivered by the power control unit through a buffer. This buffer is always enabled when the main voltage regulator of the power control unit operates in normal Run mode (refer to Reset and clock control and power control sections).

If the main voltage regulator enters low-power mode (such as Low-power run mode), this buffer is disabled and the ADC cannot be used.

### ADC Voltage regulator enable sequence

To enable the ADC voltage regulator, set ADVREGEN bit to 1 in ADC\_CR register.

### ADC voltage regulator disable sequence

To disable the ADC voltage regulator, follow the sequence below:

1. Make sure that the ADC is disabled (ADEN = 0).
2. Clear ADVREGEN bit in ADC\_CR register.

## 14.4.3 Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

Calibration should be performed before starting A/D conversion. It removes the offset error which may vary from chip to chip due to process variation.

The calibration is initiated by software by setting bit ADCAL to 1. It can be initiated only when all the following conditions are met:

- the ADC voltage regulator is enabled (ADVREGEN = 1 and LDORDY = 1),
- the ADC is disabled (ADEN = 0), and
- the Auto-off mode is disabled (AUTOFF = 0).

ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. After this, the calibration factor can be read from the ADC\_DR register (from bits 6 to 0).

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). When the ADC operating conditions change ( $V_{DDA}$  changes are the main contributor to ADC offset variations and temperature change to a lesser extend), it is recommended to re-run a calibration cycle.

The calibration factor is lost in the following cases:

- The power supply is removed from the ADC (for example when the product enters Standby mode)
- The ADC peripheral is reset.

The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

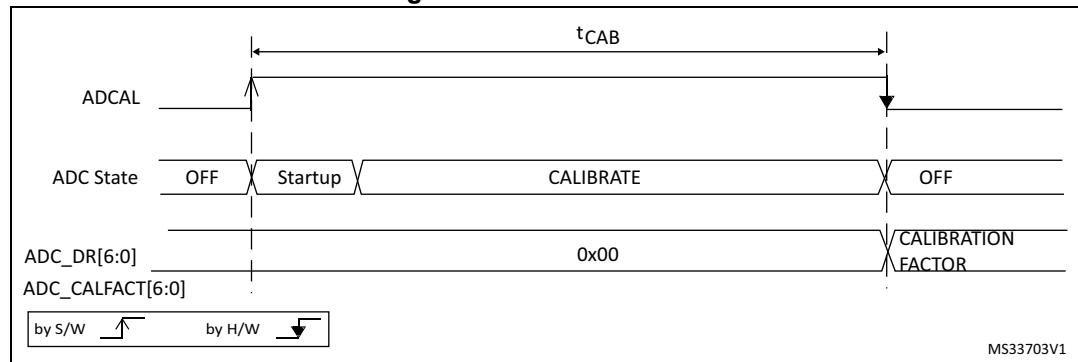
The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically

injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

#### Software calibration procedure

1. Ensure that ADEN = 0, AUTOFF = 0, ADVREGEN = 1 and DMAEN = 0.
2. Set ADCAL = 1.
3. Wait until ADCAL = 0 (or until EOCAL = 1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit in the ADC\_IER register
4. The calibration factor can be read from bits 6:0 of ADC\_DR or ADC\_CALFACT registers.
5. To reduce the noise effect of the calibration factor extraction, the software can make average of eight CALFACT[6:0] values (optional).

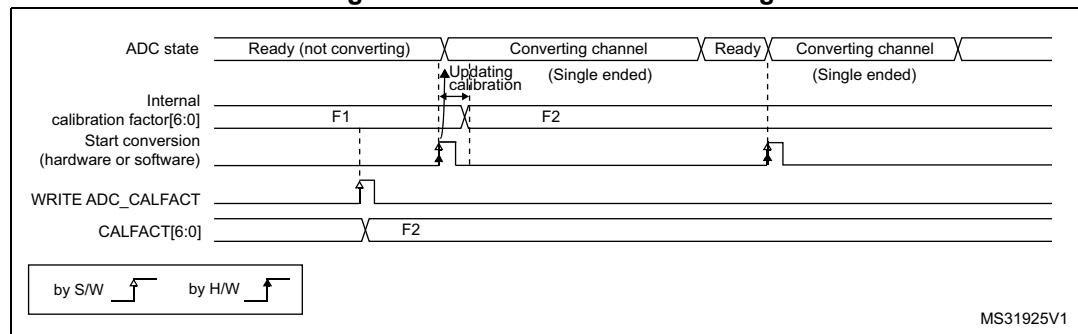
**Figure 29. ADC calibration**



#### Calibration factor forcing Software Procedure

1. Ensure that ADEN = 1 and ADSTART = 0 (ADC started with no conversion ongoing)
2. Write ADC\_CALFACT with the saved calibration factor
3. The calibration factor is used as soon as a new conversion is launched.

**Figure 30. Calibration factor forcing**



#### 14.4.4 ADC on-off control (ADEN, ADDIS, ADRDY)

At power-up, the ADC is disabled and put in power-down mode (ADEN = 0).

As shown in [Figure 31](#), the ADC needs a stabilization time of  $t_{STAB}$  before it starts converting accurately.

Two control bits are used to enable or disable the ADC:

- Set ADEN = 1 to enable the ADC. The ADRDY flag is set as soon as the ADC is ready for operation.
- Set ADDIS = 1 to disable the ADC and put the ADC in power down mode. The ADEN and ADDIS bits are then automatically cleared by hardware as soon as the ADC is fully disabled.

Conversion can then start either by setting ADSTART to 1 (refer to [Section 14.5: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN\) on page 254](#)) or when an external trigger event occurs if triggers are enabled.

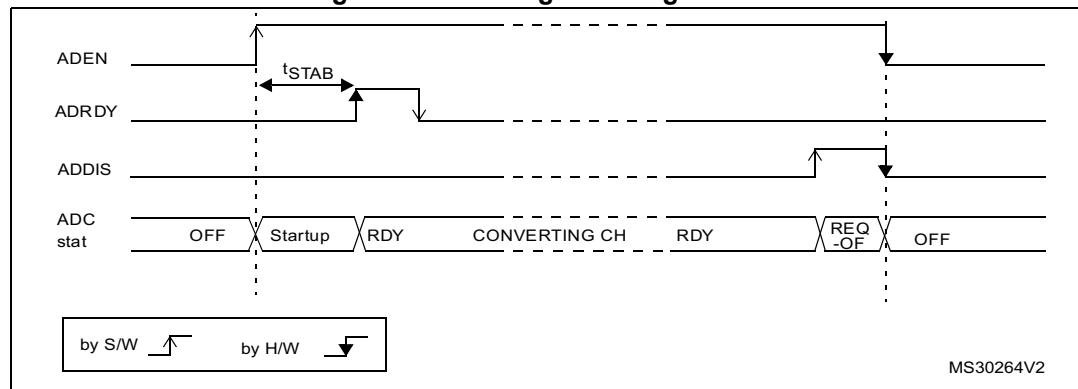
Follow this procedure to enable the ADC:

1. Clear the ADRDY bit in ADC\_ISR register by programming this bit to 1.
2. Set ADEN = 1 in the ADC\_CR register.
3. Wait until ADRDY = 1 in the ADC\_ISR register (ADRDY is set after the ADC startup time). This can be handled by interrupt if the interrupt is enabled by setting the ADRDYIE bit in the ADC\_IER register.

Follow this procedure to disable the ADC:

1. Check that ADSTART = 0 in the ADC\_CR register to ensure that no conversion is ongoing. If required, stop any ongoing conversion by writing 1 to the ADSTP bit in the ADC\_CR register and waiting until this bit is read at 0.
2. Set ADDIS = 1 in the ADC\_CR register.
3. If required by the application, wait until ADEN = 0 in the ADC\_CR register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN = 0).
4. Clear the ADRDY bit in ADC\_ISR register by programming this bit to 1 (optional).

**Figure 31. Enabling/disabling the ADC**



Note:

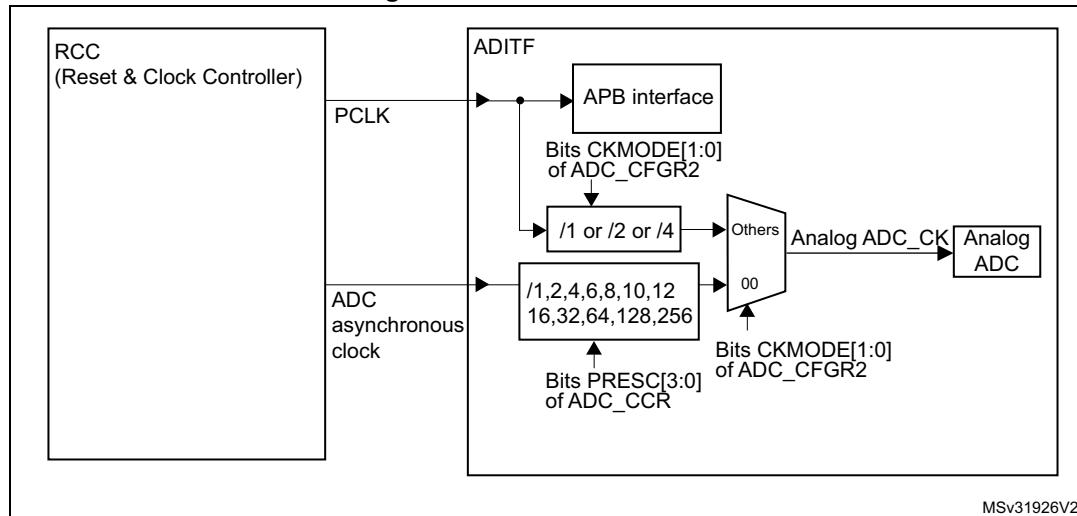
In Auto-off mode (AUTOFF = 1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.

When the bus clock is much faster than the analog ADC\_CK clock, a minimum delay of ten analog ADC\_CK cycles must be respected between ADEN and ADDIS bit settings.

#### 14.4.5 ADC clock (CKMODE, PRESC[3:0])

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC asynchronous clock) independent from the APB clock (PCLK).

**Figure 32. ADC clock scheme**



MSv31926V2

1. Refer to *Section Reset and clock control (RCC)* for how the PCLK clock and ADC asynchronous clock are enabled.

The input clock of the analog ADC can be selected between two different clock sources (see [Figure 32: ADC clock scheme](#) to see how the PCLK clock and the ADC asynchronous clock are enabled):

- a) The ADC clock can be a specific clock source, named “ADC asynchronous clock” which is independent and asynchronous with the APB clock.  
Refer to RCC Section for more information on generating this clock source.  
To select this scheme, bits CKMODE[1:0] of the ADC\_CFGR2 register must be reset.
- b) The ADC clock can be derived from the APB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4) according to bits CKMODE[1:0].  
To select this scheme, bits CKMODE[1:0] of the ADC\_CFGR2 register must be different from “00”.

In option a), the generated ADC clock can eventually be divided by a prescaler (1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256) when programming the bits PRESC[3:0] in the ADC\_CCR register).

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the APB clock scheme selected.

Option b) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

**Table 54. Latency between trigger and start of conversion<sup>(1)</sup>**

ADC clock source	CKMODE[1:0]	Latency between the trigger event and the start of conversion
SYSCLK, or HSIKER clock <sup>(2)</sup>	00	Latency is not deterministic (jitter)
PCLK divided by 2	01	Latency is deterministic (no jitter) and equal to 3.25 ADC clock cycles
PCLK divided by 4	10	Latency is deterministic (no jitter) and equal to 3.125 ADC clock cycles
PCLK divided by 1	11	Latency is deterministic (no jitter) and equal to 3 ADC clock cycles

1. Refer to the device datasheet for the maximum ADC\_CLK frequency.

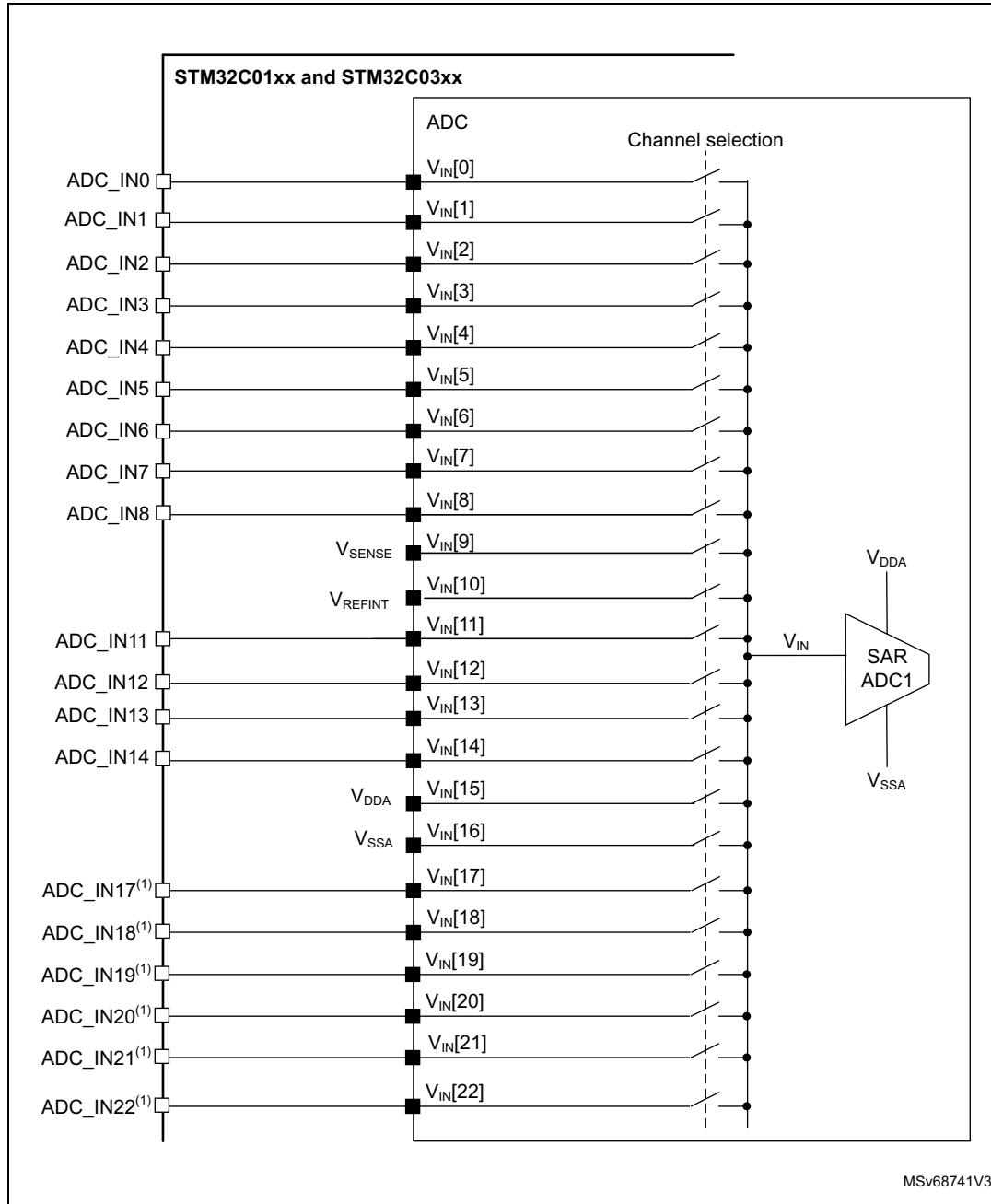
2. Selected with ADCSEL bitfield of the RCC\_CCIPR register.

**Caution:** When selecting CKMODE[1:0] = 11 (PCLK divided by 1), the user must ensure that the PCLK has a 50% duty cycle. This is done by selecting a system clock with a 50% duty cycle and configuring the APB prescaler in bypass modes in the RCC (refer to there Reset and clock controller section). If an internal source clock is selected, the AHB and APB prescalers do not divide the clock.

#### 14.4.6 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in [Figure 33](#).

**Figure 33. ADC connectivity**



1. ADC\_IN17 to ADC\_IN22 are available only on STM32C03xx devices.

#### 14.4.7 Configuring the ADC

The software must write the ADCAL and ADEN bits in the ADC\_CR register and configure the ADC\_CFGR1 and ADC\_CFGR2 registers only when the ADC is disabled (ADEN must be cleared).

The software must only write to the ADSTART and ADDIS bits in the ADC\_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC\_IER, ADC\_SMPR, ADC\_CHSELR and ADC\_CCR registers, refer to the description of the corresponding control bit in [Section 14.12: ADC registers](#).

ADC\_AWDTRx registers can be modified when conversion is ongoing.

The software must only write to the ADSTP bit in the ADC\_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

**Note:**

*There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled (clear ADEN = 0 and all the bits in the ADC\_CR register).*

#### 14.4.8 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 23 multiplexed channels:

- Up to 19 analog inputs from GPIO pins (ADC\_INx)
- 4 internal analog inputs (temperature sensor, internal reference voltage, analog supply and analog ground)

It is possible to convert a single channel or a sequence of channels.

The sequence of the channels to be converted can be programmed in the ADC\_CHSELR channel selection register: each analog input channel has a dedicated selection bit (CHSELx).

The ADC scan sequencer can be used in two different modes:

- Sequencer not fully configurable:

The order in which the channels are scanned is defined by the channel number (CHSELRMOD bit must be cleared in ADC\_CFGR1 register):

- Sequence length configured through CHSELx bits in ADC\_CHSELR register
- Sequence direction: the channels are scanned in a forward direction (from the lowest to the highest channel number) or backward direction (from the highest to the lowest channel number) depending on the value of SCANDIR bit (SCANDIR = 0: forward scan, SCANDIR = 1: backward scan)

- Any channel can belong to in these sequences
- Sequencer fully configurable
  - The CHSELRMOD bit is set in ADC\_CFGR1 register.
  - Sequencer length is up to 8 channels
  - The order in which the channels are scanned is independent from the channel number. Any order can be configured through SQ1[3:0] to SQ8[3:0] bits in ADC\_CHSELR register.
  - Only channel 0 to channel 14 can be selected in this sequence
  - If the sequencer detects SQx[3:0] = 0b1111, the following SQx[3:0] registers are ignored.
  - If no 0b1111 is programmed in SQx[3:0], the sequencer scans full eight channels.

After programming ADC CHSELR, SCANDIR and CHSELRMOD bits, it is mandatory to wait for CCRDY flag before starting conversions. It indicates that the new channel setting has been applied. If a new configuration is required, the CCRDY flag must be cleared prior to starting the conversion.

The software is allowed to program the CHSEL, SCANDIR, CHSELRMOD bits only when ADSTART bit is cleared (which ensures that no conversion is ongoing).

#### **Temperature sensor, V<sub>REFINT</sub> and V<sub>DDA</sub> and V<sub>SSA</sub> internal channels**

The temperature sensor is connected to channel ADC V<sub>IN</sub>[9].

The internal voltage reference V<sub>REFINT</sub> is connected to channel ADC V<sub>IN</sub>[10].

V<sub>DDA</sub> channel is connected to ADC V<sub>IN</sub>[15].

V<sub>SSA</sub> channel is connected to ADC V<sub>IN</sub>[16].

#### **14.4.9 Programmable sampling time (SMPx[2:0])**

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows the conversion speed to be trimmed according to the input resistance of the input voltage source.

The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP1[2:0] and SMP2[2:0] bits in the ADC\_SMPR register.

Each channel can choose one out of two sampling times configured in SMP1[2:0] and SMP2[2:0] bitfields, through SMPSELx bits in ADC\_SMPR register.

The total conversion time is calculated as follows:

$$t_{\text{CONV}} = \text{Sampling time} + 12.5 \times \text{ADC clock cycles}$$

Example:

With ADC\_CLK = 16 MHz and a sampling time of 1.5 ADC clock cycles:

$$t_{\text{CONV}} = 1.5 + 12.5 = 14 \text{ ADC clock cycles} = 0.875 \mu\text{s}$$

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

#### 14.4.10 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT = 0 in the ADC\_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC\_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC\_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

*Note:* To convert a single channel, program a sequence with a length of 1.

#### 14.4.11 Continuous conversion mode (CONT = 1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT = 1 in the ADC\_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC\_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC\_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

*Note:* To convert a single channel, program a sequence with a length of 1.

*It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.*

#### 14.4.12 Starting conversions (ADSTART)

Software starts ADC conversions by setting ADSTART = 1.

When ADSTART is set, the conversion:

- Starts immediately if EXTN = 00 (software trigger)
- At the next active edge of the selected hardware trigger if EXTN ≠ 00

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:

- In single mode with software trigger (CONT = 0, EXTN = 00)
  - At any end of conversion sequence (EOS = 1)
- In discontinuous mode with software trigger (CONT = 0, DISCEN = 1, EXTN = 00)
  - At end of conversion (EOC = 1)
- In all cases (CONT = x, EXTN = XX)
  - After execution of the ADSTP procedure invoked by software (see [Section 14.4.14: Stopping an ongoing conversion \(ADSTP\) on page 254](#))

*Note: In continuous mode (CONT = 1), the ADSTART bit is not cleared by hardware when the EOS flag is set because the sequence is automatically relaunched.*

*When hardware trigger is selected in single mode (CONT = 0 and EXTN = 01), ADSTART is not cleared by hardware when the EOS flag is set (except if DMAEN = 1 and DMACFG = 0 in which case ADSTART is cleared at end of the DMA transfer). This avoids the need for software having to set the ADSTART bit again and ensures the next trigger event is not missed.*

*After changing channel selection configuration (by programming ADC\_CHSEL register or changing CHSELRMOD or SCANDIR), it is mandatory to wait until CCRDY flag is asserted before asserting ADSTART, otherwise the value written to ADSTART is ignored.*

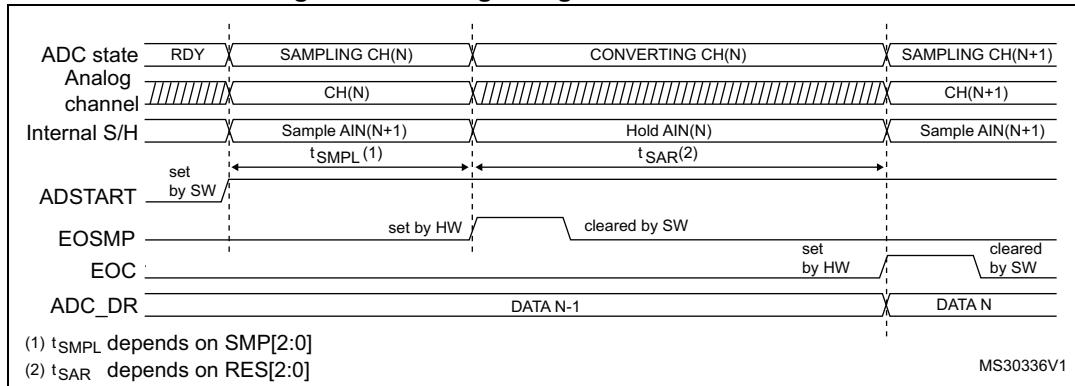
### 14.4.13 Timings

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

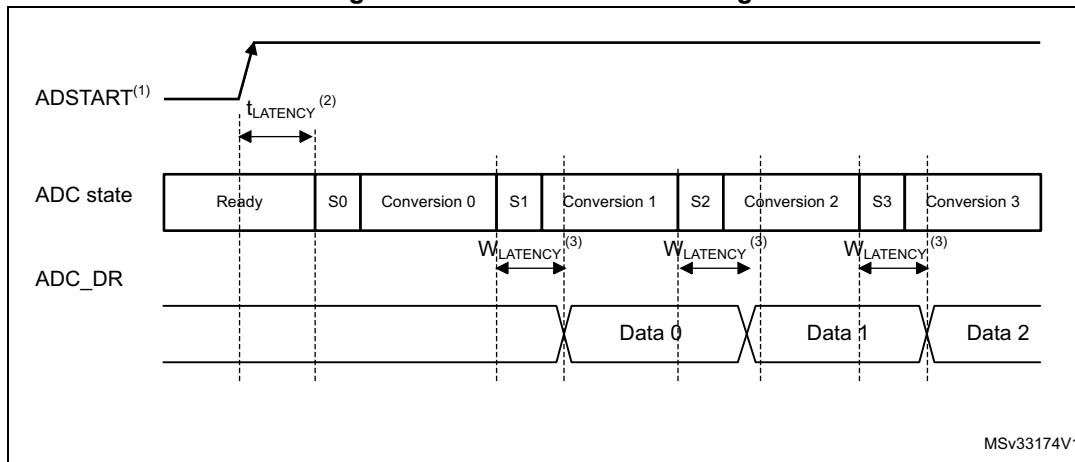
$$t_{\text{CONV}} = t_{\text{SMPL}} + t_{\text{SAR}} = [1.5 \text{ } t_{\text{min}} + 12.5 \text{ } t_{\text{12bit}}] \times t_{\text{ADC\_CLK}}$$

$$t_{\text{CONV}} = t_{\text{SMPL}} + t_{\text{SAR}} = 42.9 \text{ ns } t_{\text{min}} + 357.1 \text{ ns } t_{\text{12bit}} = 0.400 \mu\text{s } t_{\text{min}} \text{ (for } f_{\text{ADC\_CLK}} = 35 \text{ MHz)}$$

**Figure 34. Analog to digital conversion time**



**Figure 35. ADC conversion timings**



1. EXTEN = 00 or EXTEN ≠ 00
2. Trigger latency (refer to datasheet for more details)
3. ADC\_DR register write latency (refer to datasheet for more details)

#### 14.4.14 Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP = 1 in the ADC\_CR register.

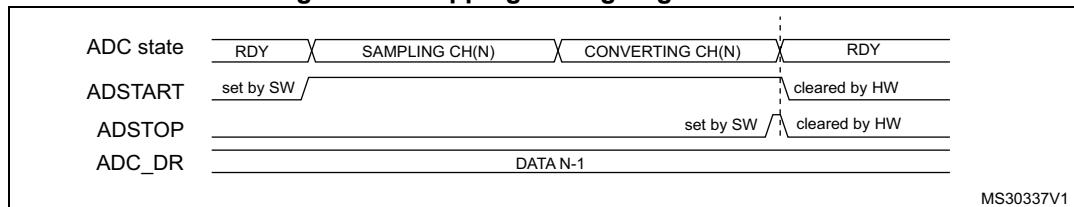
This resets the ADC operation and the ADC is idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (ADC\_DR register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that restarting the ADC would restart a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware and the software must wait until ADSTART=0 before starting new conversions.

**Figure 36. Stopping an ongoing conversion**



#### 14.5 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

If bit ADSTART = 0, any hardware triggers which occur are ignored.

**Table 55** provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

**Table 55. Configuring the trigger polarity**

Source	EXTEN[1:0]
Trigger detection disabled	00
Detection on rising edge	01
Detection on falling edge	10
Detection on both rising and falling edges	11

**Note:** The polarity of the external trigger can be changed only when the ADC is not converting (ADSTART = 0).

The EXTSEL[2:0] control bits are used to select which of 8 possible events can trigger conversions.

Refer to [Table 53: External triggers](#) in [Section 14.4.1: ADC pins and internal signals](#) for the list of all the external triggers that can be used for regular conversion.

The software source trigger events can be generated by setting the ADSTART bit in the ADC\_CR register.

*Note:* *The trigger selection can be changed only when the ADC is not converting (ADSTART = 0).*

#### 14.5.1 Discontinuous mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC\_CFGR1 register.

In this mode (DISCEN = 1), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if DISCEN = 0, a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:

- DISCEN = 1, channels to be converted = 0, 3, 7, 10
  - 1st trigger: channel 0 is converted and an EOC event is generated
  - 2nd trigger: channel 3 is converted and an EOC event is generated
  - 3rd trigger: channel 7 is converted and an EOC event is generated
  - 4th trigger: channel 10 is converted and both EOC and EOS events are generated.
  - 5th trigger: channel 0 is converted an EOC event is generated
  - 6th trigger: channel 3 is converted and an EOC event is generated
  - ...
- DISCEN = 0, channels to be converted = 0, 3, 7, 10
  - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOS event.
  - Any subsequent trigger events restarts the complete sequence.

*Note:* *It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.*

#### 14.5.2 Programmable resolution (RES) - Fast conversion mode

It is possible to obtain faster conversion times ( $t_{SAR}$ ) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC\_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

*Note:* *The RES[1:0] bit must only be changed when the ADEN bit is reset.*

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeros.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in [Table 56](#).

**Table 56.  $t_{SAR}$  timings depending on resolution**

<b>RES[1:0] bits</b>	<b><math>t_{SAR}</math> (ADC clock cycles)</b>	<b><math>t_{SAR}</math> (ns) at <math>f_{ADC} = 35</math> MHz</b>	<b><math>t_{SMPL}</math> (min) (ADC clock cycles)</b>	<b><math>t_{CONV}</math> (ADC clock cycles) (with min. <math>t_{SMPL}</math>)</b>	<b><math>t_{CONV}</math> (ns) at <math>f_{ADC} = 35</math> MHz</b>
12	12.5	357	1.5	14	400
10	10.5	300	1.5	12	343
8	8.5	243	1.5	10	286
6	6.5	186	1.5	8	229

#### 14.5.3 End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC\_ISR register as soon as a new conversion data result is available in the ADC\_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC\_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC\_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC\_ISR register. The EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC\_IER register.

The aim of this interrupt is to allow the processing to be synchronized with the conversions. Typically, an analog multiplexer can be accessed in hidden time during the conversion phase, so that the multiplexer is positioned when the next sampling starts.

**Note:** *As there is only a very short time left between the end of the sampling and the end of the conversion, it is recommended to use polling or a WFE instruction rather than an interrupt and a WFI instruction.*

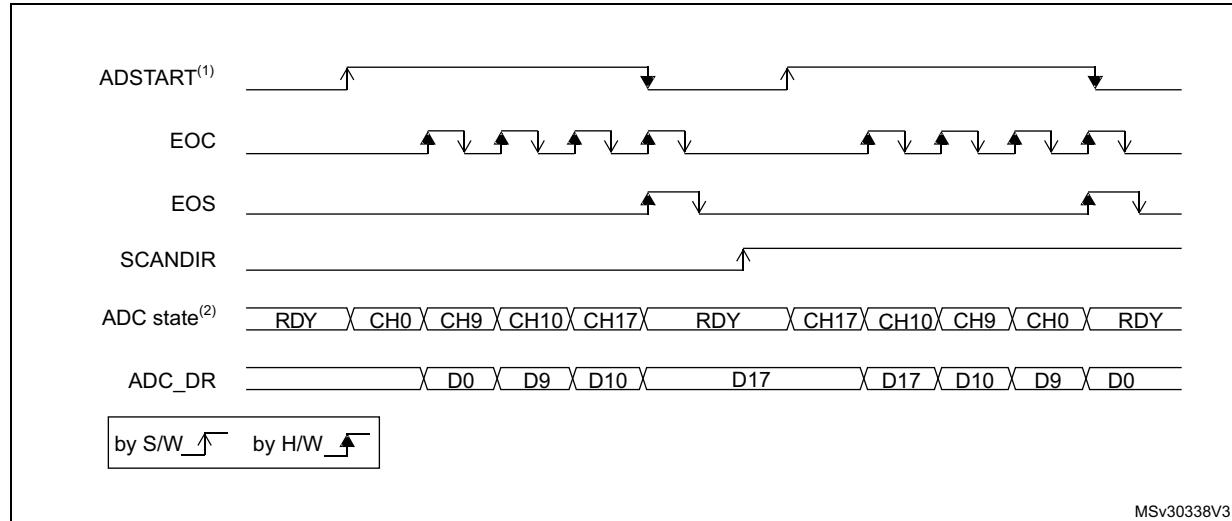
#### 14.5.4 End of conversion sequence (EOS flag)

The ADC notifies the application of each end of sequence (EOS) event.

The ADC sets the EOS flag in the ADC\_ISR register as soon as the last data result of a conversion sequence is available in the ADC\_DR register. An interrupt can be generated if the EOSIE bit is set in the ADC\_IER register. The EOS flag is cleared by software by writing 1 to it.

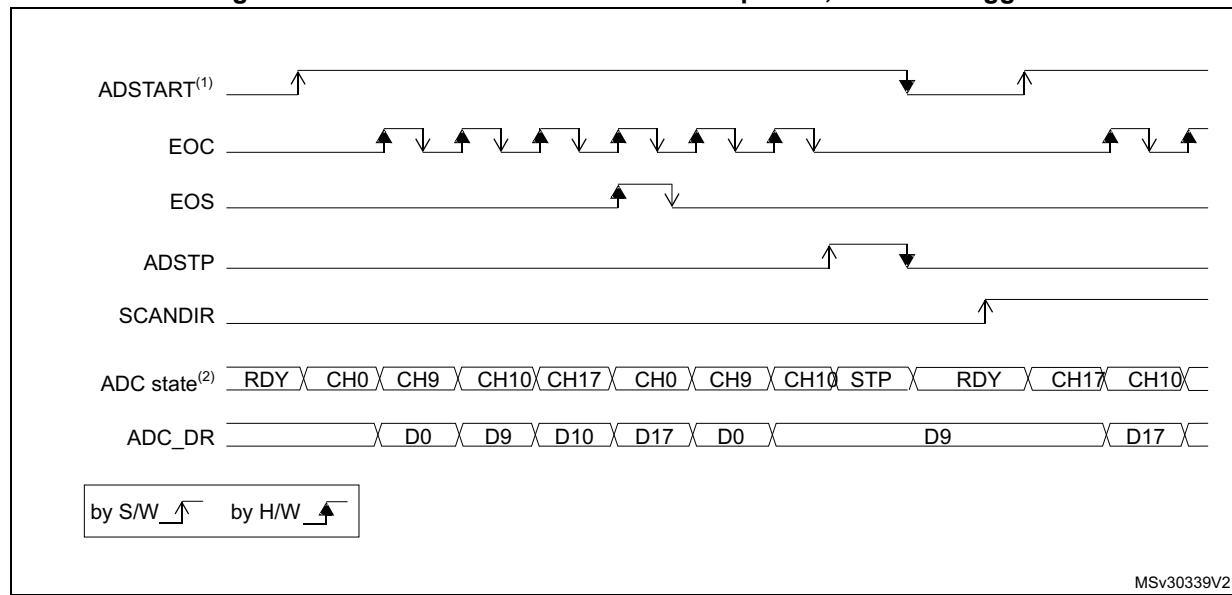
### 14.5.5 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 37. Single conversions of a sequence, software trigger

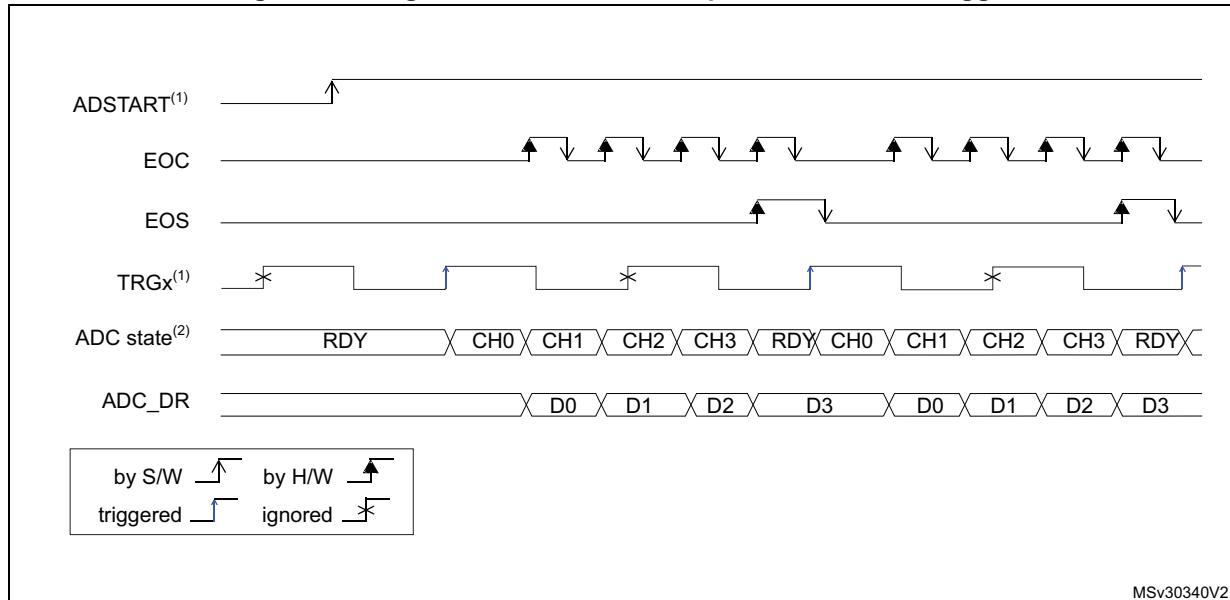


1. EXTEN = 00, CONT = 0
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

Figure 38. Continuous conversion of a sequence, software trigger



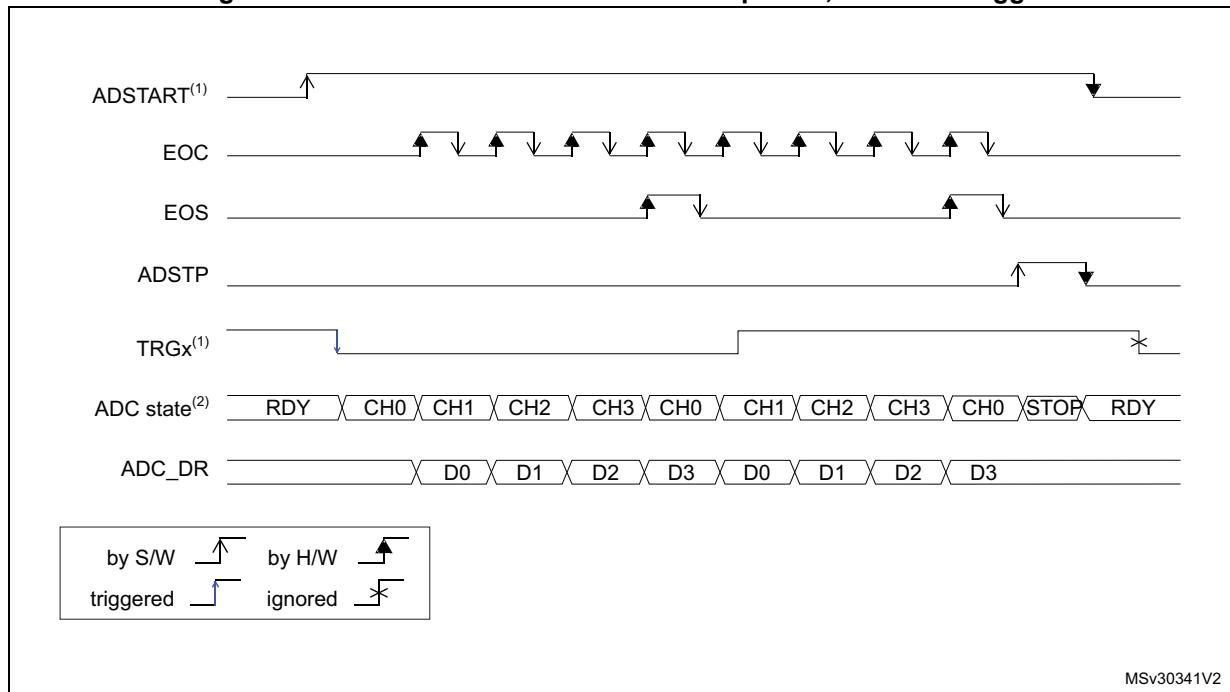
1. EXTEN = 00, CONT = 1,
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

**Figure 39. Single conversions of a sequence, hardware trigger**

MSv30340V2

1. EXTSEL = TRGx (over-frequency), EXTEN = 01 (rising edge), CONT = 0

2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

**Figure 40. Continuous conversions of a sequence, hardware trigger**

MSv30341V2

1. EXTSEL = TRGx, EXTEN = 10 (falling edge), CONT = 1

2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

### 14.5.6 Low frequency trigger mode

Once the ADC is enabled or the last ADC conversion is complete, the ADC is ready to start a new conversion. The ADC needs to be started at a predefined time ( $t_{idle}$ ) otherwise ADC converted data might be corrupted due to the transistor leakage (refer to the device datasheet for the maximum value of  $t_{idle}$ ).

If the application has to support a time longer than the maximum  $t_{idle}$  value (between one trigger to another for single conversion mode or between the ADC enable and the first ADC conversion), then the ADC internal state needs to be rearmed. This mechanism can be enabled by setting LFTRIG bit to 1 in ADC\_CFGR2 register. By setting this bit, any trigger (software or hardware) sends a rearm command to ADC. The conversion starts after a one ADC clock cycle delay compared to LFTRIG cleared.

It is not necessary to use this mode when AUTOFF bit is set. For Wait mode, only the first trigger generates an internal rearm command.

## 14.6 Data management

### 14.6.1 Data register and data alignment (ADC\_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC\_DR data register which is 16-bit wide.

The format of the ADC\_DR depends on the configured data alignment and resolution.

The ALIGN bit in the ADC\_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN = 0) or left-aligned (ALIGN = 1) as shown in [Figure 41](#).

**Figure 41. Data alignment and resolution (oversampling disabled: OVSE = 0)**

ALIGN	RES	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0x0	0x0															DR[11:0]
	0x1		0x00														DR[9:0]
	0x2			0x00													DR[7:0]
	0x3				0x00												DR[5:0]
1	0x0					DR[11:0]											0x0
	0x1					DR[9:0]											0x00
	0x2					DR[7:0]											0x00
	0x3					0x00					DR[5:0]						0x0

MS30342V1

### 14.6.2 ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the data from a new conversion is available.

The OVR flag is set in the ADC\_ISR register if the EOC flag is still at '1' at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC\_IER register.

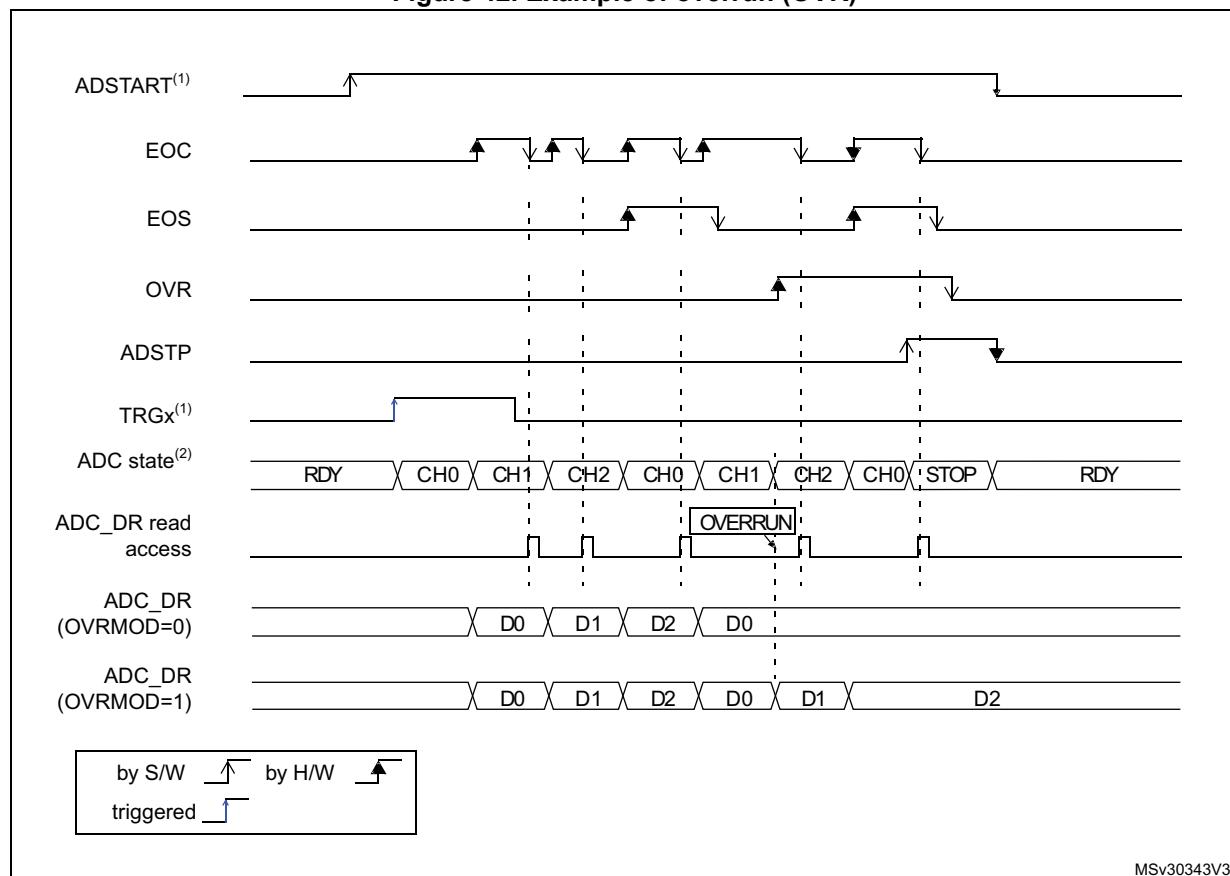
When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC\_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVREMOD bit in the ADC\_CFGR1 register:

- OVREMOD = 0
  - An overrun event preserves the data register from being overwritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is discarded.
- OVREMOD = 1
  - The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, further conversions can be performed and the ADC\_DR register always contains the data from the latest conversion.

**Figure 42. Example of overrun (OVR)**



### 14.6.3 Managing a sequence of data converted without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC\_ISR register and the ADC\_DR register can be read. The OVRMOD bit in the ADC\_CFGR1 register should be configured to 0 to manage overrun events as an error.

### 14.6.4 Managing converted data without using the DMA without overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag should be ignored by the software. When OVRMOD = 1, an overrun event does not prevent the ADC from continuing to convert and the ADC\_DR register always contains the latest conversion data.

### 14.6.5 Managing converted data using the DMA

Since all converted channel values are stored in a single data register, it is efficient to use DMA when converting more than one channel. This avoids losing the conversion data results stored in the ADC\_DR register.

When DMA mode is enabled (DMAEN bit set in the ADC\_CFGR1 register), a DMA request is generated after the conversion of each channel. This allows the transfer of the converted data from the ADC\_DR register to the destination location selected by the software.

*Note:* The DMAEN bit in the ADC\_CFGR1 register must be set after the ADC calibration phase.

Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to [Section 14.6.2: ADC overrun \(OVR, OVRMOD\) on page 259](#)).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG in the ADC\_CFGR1 register:

- DMA one shot mode (DMACFG = 0).  
This mode should be selected when the DMA is programmed to transfer a fixed number of data words.
- DMA circular mode (DMACFG = 1)  
This mode should be selected when programming the DMA in circular mode or double buffer mode.

#### DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs, see [Section 9: Direct memory access controller \(DMA\) on page 182](#)) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted and its partial result discarded
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- The scan sequence is stopped and reset
- The DMA is stopped

#### DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available in the data register, even if the DMA has reached the last DMA transfer. This allows the DMA to be configured in circular mode to handle a continuous analog input data stream.

## 14.7 Low-power features

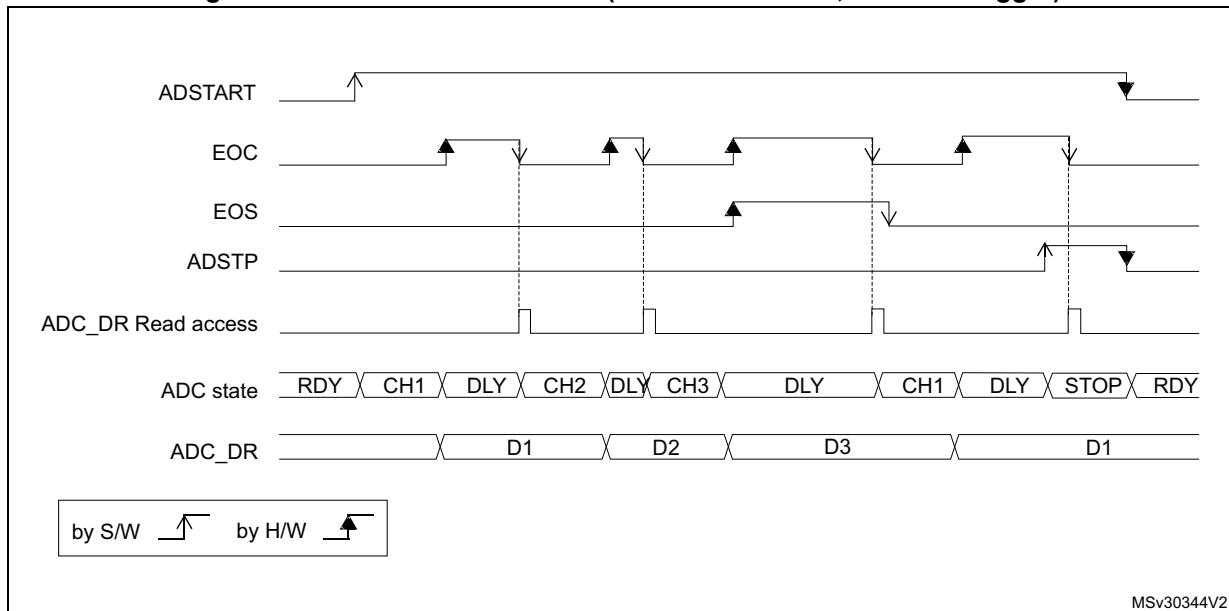
### 14.7.1 Wait mode conversion

Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set in the ADC\_CFGR1 register, a new conversion can start only if the previous data has been treated, once the ADC\_DR register has been read or if the EOC bit has been cleared.

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

**Note:** *Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.*

**Figure 43. Wait mode conversion (continuous mode, software trigger)**

1. EXTEN = 00, CONT = 1
2. CHSEL = 0x3, SCANDIR = 0, WAIT = 1, AUTOFF = 0

#### 14.7.2 Auto-off mode (AUTOFF)

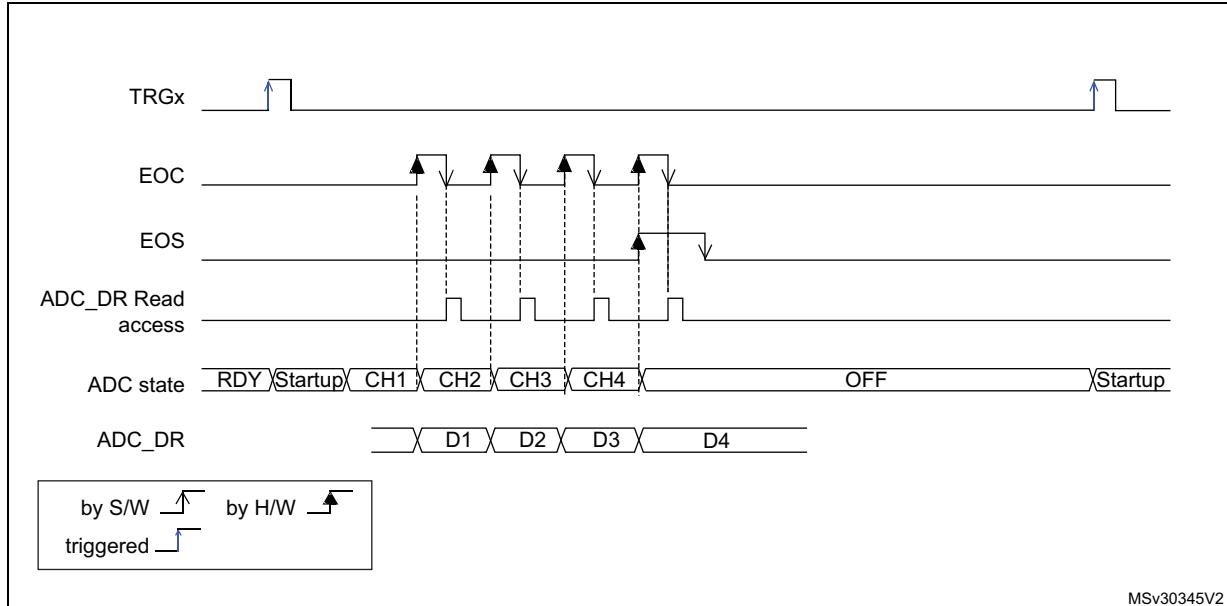
The ADC has an automatic power management feature which is called auto-off mode, and is enabled by setting AUTOFF = 1 in the ADC\_CFGR1 register.

When AUTOFF = 1, the ADC is always powered off when not converting and automatically wakes-up when a conversion is started (by software or hardware trigger). A startup-time is automatically inserted between the trigger event which starts the conversion and the sampling time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete.

Auto-off mode can cause a dramatic reduction in the power consumption of applications which need relatively few conversions or when conversion requests are timed far enough apart (for example with a low frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

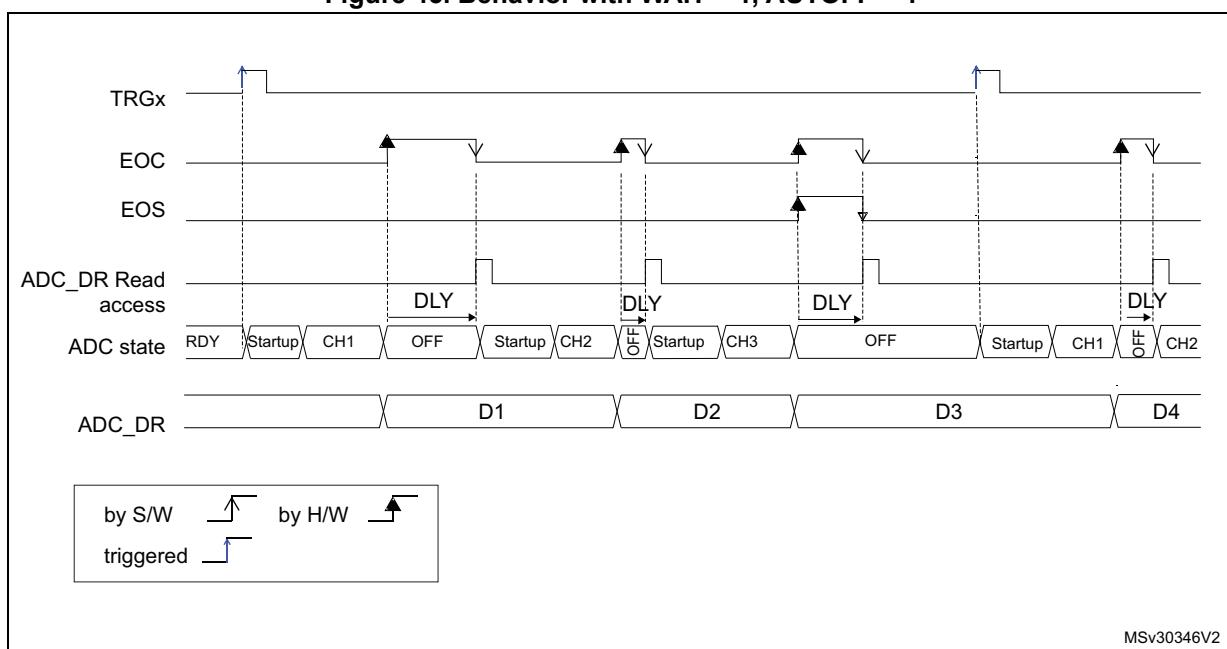
Auto-off mode can be combined with the wait mode conversion (WAIT = 1) for applications clocked at low frequency. This combination can provide significant power savings if the ADC is automatically powered-off during the wait phase and restarted as soon as the ADC\_DR register is read by the application (see [Figure 45: Behavior with WAIT = 1, AUTOFF = 1](#)).

**Figure 44. Behavior with WAIT = 0, AUTOFF = 1**



- EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1

**Figure 45. Behavior with WAIT = 1, AUTOFF = 1**



- EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1

## 14.8 Analog window watchdogs

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

### 14.8.1 Description of analog watchdog 1

AWD1 analog watchdog is enabled by setting the AWD1EN bit in the ADC\_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels (see [Table 58: Analog watchdog 1 channel selection](#)) remain within a configured voltage range (window) as shown in [Figure 46](#).

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in HTx[11:0] and LTx[11:0] bits of ADC\_AWD1TR register. An interrupt can be enabled by setting the AWD1IE bit in the ADC\_IER register.

The AWD1 flag is cleared by software by programming it to 1.

When converting data with a resolution of less than 12-bit (according to bits DRES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

[Table 57](#) describes how the comparison is performed for all the possible resolutions.

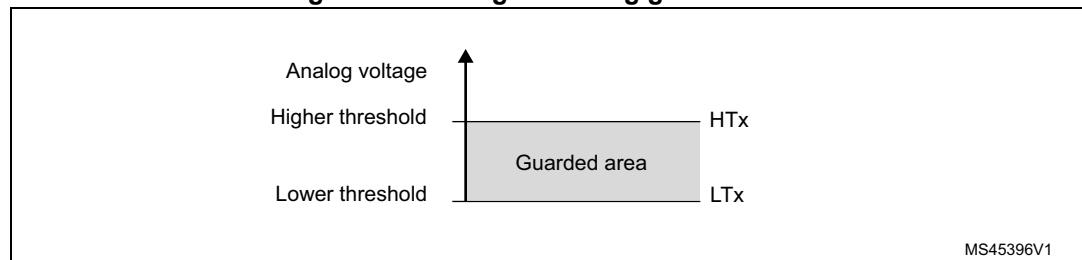
**Table 57. Analog watchdog comparison**

Resolution bits RES[1:0]	Analog Watchdog comparison between:		Comments
	Raw converted data, left aligned <sup>(1)</sup>	Thresholds	
00: 12-bit	DATA[11:0]	LTx[11:0] and HTx[11:0]	-
01: 10-bit	DATA[11:2],00	LTx[11:0] and HTx[11:0]	The user must configure LTx[1:0] and HTx[1:0] to “00”
10: 8-bit	DATA[11:4],0000	LTx[11:0] and HTx[11:0]	The user must configure LTx[3:0] and HTx[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	LTx[11:0] and HTx[11:0]	The user must configure LTx[5:0] and HTx[5:0] to “000000”

1. The watchdog comparison is performed on the raw converted data before any alignment calculation.

[Table 58](#) shows how to configure the AWD1SGL and AWD1EN bits in the ADC\_CFGR1 register to enable the analog watchdog on one or more channels.

**Figure 46. Analog watchdog guarded area**



**Table 58. Analog watchdog 1 channel selection**

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit
None	x	0
All channels	0	1
Single <sup>(1)</sup> channel	1	1

1. Selected by the AWD1CH[4:0] bits

#### 14.8.2 Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the AWDxCHy in ADC\_AWDxCR ( $x = 2, 3$ ).

The corresponding watchdog is enabled when any AWDxCHy bit ( $x = 2, 3$ ) is set in ADC\_AWDxCR register.

When converting data with a resolution of less than 12 bits (configured through DRES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

*Table 57* describes how the comparison is performed for all the possible resolutions.

The AWD2/3 analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in HTx[11:0] and LTx[11:0] of ADC\_AWDxTR registers ( $x = 2$  or  $3$ ). An interrupt can be enabled by setting the AWDxIE bit in the ADC\_IER register.

The AWD2 and ADW3 flags are cleared by software by programming them to 1.

#### 14.8.3 ADC\_AWDx\_OUT output signal generation

Each analog watchdog is associated to an internal hardware signal, ADC\_AWDx\_OUT ( $x$  being the watchdog number) that is directly connected to the ETR input (external trigger) of some on-chip timers (refer to the timers section for details on how to select the ADC\_AWDx\_OUT signal as ETR).

ADC\_AWDx\_OUT is activated when the associated analog watchdog is enabled:

- ADC\_AWDx\_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC\_AWDx\_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds. It remains at 1 if the next guarded conversions are still outside the programmed thresholds.
- ADC\_AWDx\_OUT is also reset when disabling the ADC (when setting ADDIS to 1). Note that stopping conversions (ADSTP set), might clear the ADC\_AWDx\_OUT state.
- ADC\_AWDx\_OUT state does not change when the ADC converts the none-guarded channel (see *Figure 49*)

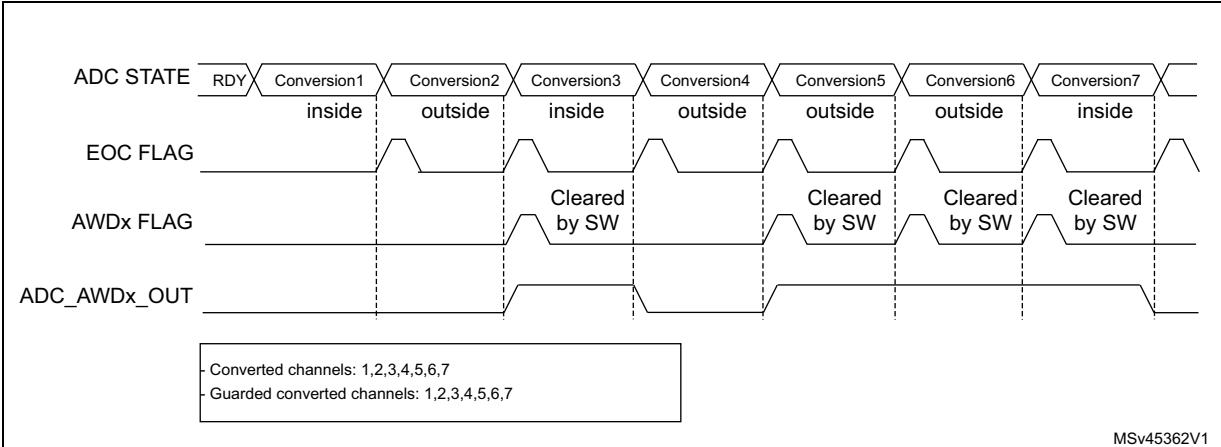
AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADC\_AWDx\_OUT (as an example, ADC\_AWDx\_OUT can toggle while AWDx flag remains at 1 if the software has not cleared the flag).

The ADC\_AWDx\_OUT signal is generated by the ADC\_CLK domain. This signal can be generated even the APB clock is stopped.

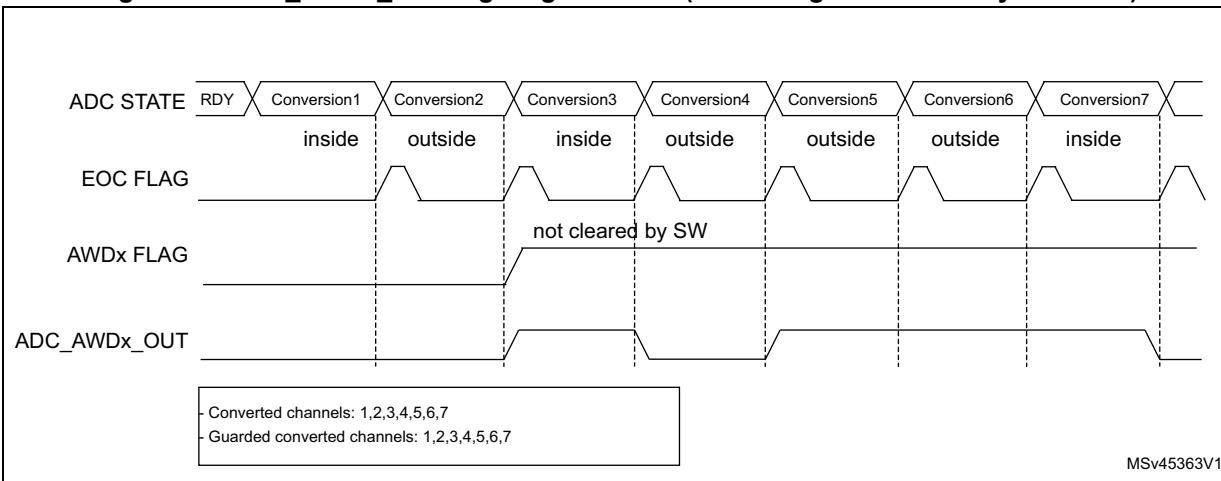
The AWD comparison is performed at the end of each ADC conversion. The ADC\_AWDx\_OUT rising edge and falling edge occurs two ADC\_CLK clock cycles after the comparison.

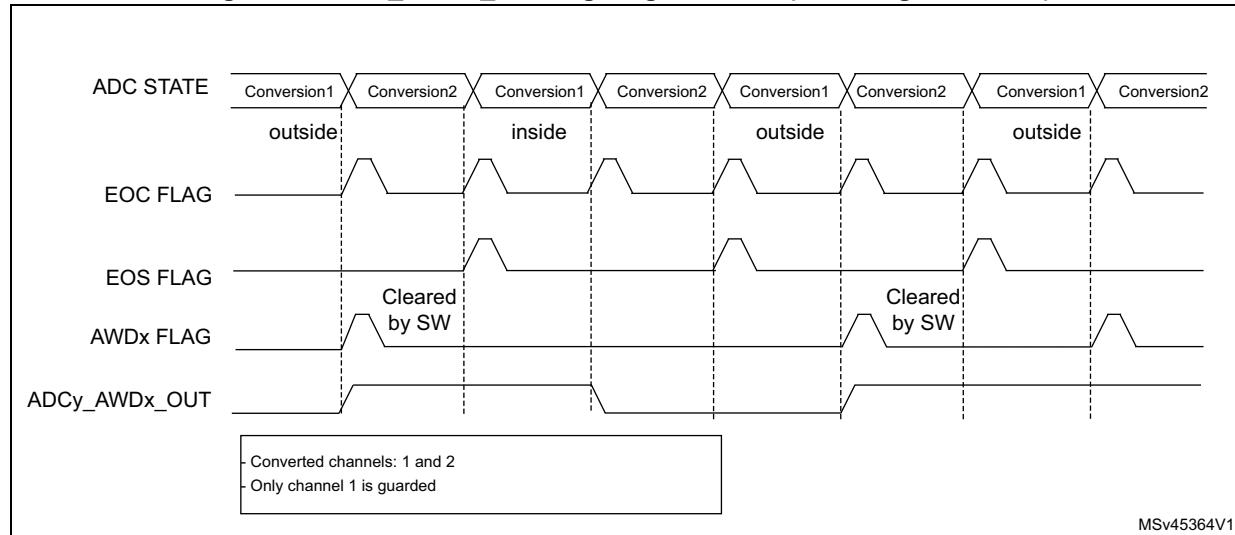
As ADC\_AWDx\_OUT is generated by the ADC\_CLK domain and AWD flag is generated by the APB clock domain, the rising edges of these signals are not synchronized.

**Figure 47. ADC\_AWDx\_OUT signal generation**



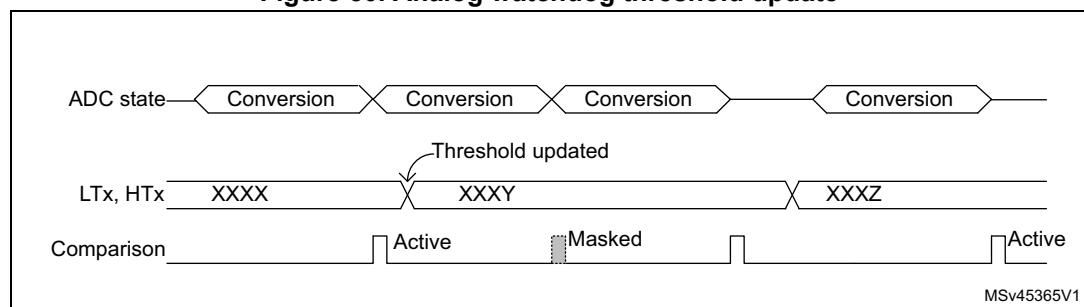
**Figure 48. ADC\_AWDx\_OUT signal generation (AWDx flag not cleared by software)**



**Figure 49. ADC\_AWDx\_OUT signal generation (on a single channel)**

#### 14.8.4 Analog Watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed during an analog-to-digital conversion (that is between the start of the conversion and the end of conversion of the ADC internal state). If HTx and LTx bits are programmed during the ADC guarded channel conversion, the watchdog function is masked for this conversion. This mask is cleared when starting a new conversion, and the resulting new AWD threshold is applied starting the next ADC conversion result. AWD comparison is performed at each end of conversion. If the current ADC data are out of the new threshold interval, this does not generate any interrupt or an ADC\_AWDx\_OUT signal. The Interrupt and the ADC\_AWDx\_OUT generation only occurs at the end of the ADC conversion that started after the threshold update. If ADC\_AWDx\_OUT is already asserted, programming the new threshold does not deassert the ADC\_AWDx\_OUT signal.

**Figure 50. Analog watchdog threshold update**

## 14.9 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{N-1} \text{Conversion}(t_n)$$

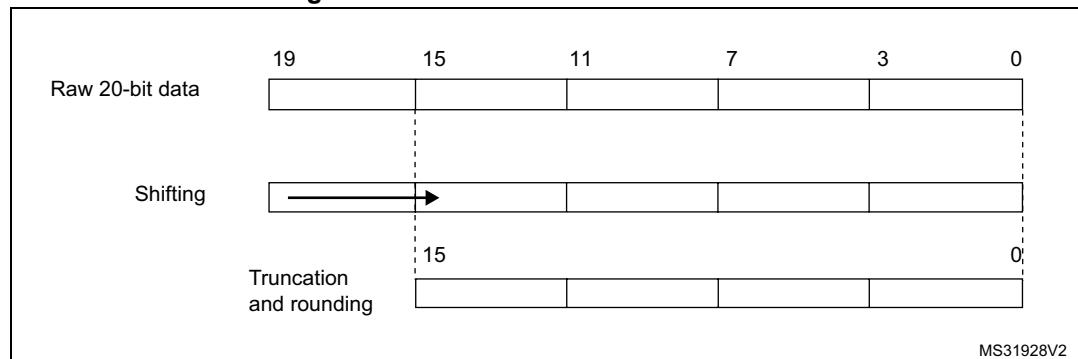
It allows the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC\_CFGR2 register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC\_CFGR2 register.

The summation unit can yield a result up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC\_DR data register.

**Note:** *If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.*

**Figure 51. 20-bit to 16-bit result truncation**



The [Figure 52](#) gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 52. Numerical example with 5-bits shift and rounding**

Raw 20-bit data:	19	15	11	7	3	
		3	B	7	D	7
Final result after 5-bits shift and rounding to nearest	15				0	
		1	D	B	F	

MS31929V1

The [Table 59](#) below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFFF.

**Table 59. Maximum output results vs N and M. Grayed values indicates truncation**

Oversampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFFF0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFFF0	0xFF00	0xFF80	0xFFC0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800

The conversion timings in oversampled mode do not change compared to standard conversion mode: the sample time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to  $N \times t_{CONV} = N \times (t_{SMPL} + t_{SAR})$ . The flags features are raised as following:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- the end of sequence (EOCSEQ) occurs once the sequence of oversampled data is completed (i.e. after  $N \times$  sequence length conversions total)

### 14.9.1 ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are available:

- Single or continuous mode conversions, forward or backward scanned sequences
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (WAIT, AUTOFF)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC\_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

**Note:** *The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC\_CFGR1 is ignored and the data are always provided right-aligned.*

### 14.9.2 Analog watchdog

The analog watchdog functionality is available, with the following differences:

- the RES[1:0] bits are ignored, comparison is always done on using the full 12-bits values HTx[11:0] and LTx[11:0]
- the comparison is performed on the most significant 12 bits of the 16 bits oversampled results ADC\_DR[15:4]

**Note:** *Care must be taken when using high shifting values. This reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits thus yielding a 12-bit data right-aligned, the affective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC\_DR[11:4] and HTx[7:0] / LTx[7:0], and HTx[11:8] / LTx[11:8] must be kept reset.*

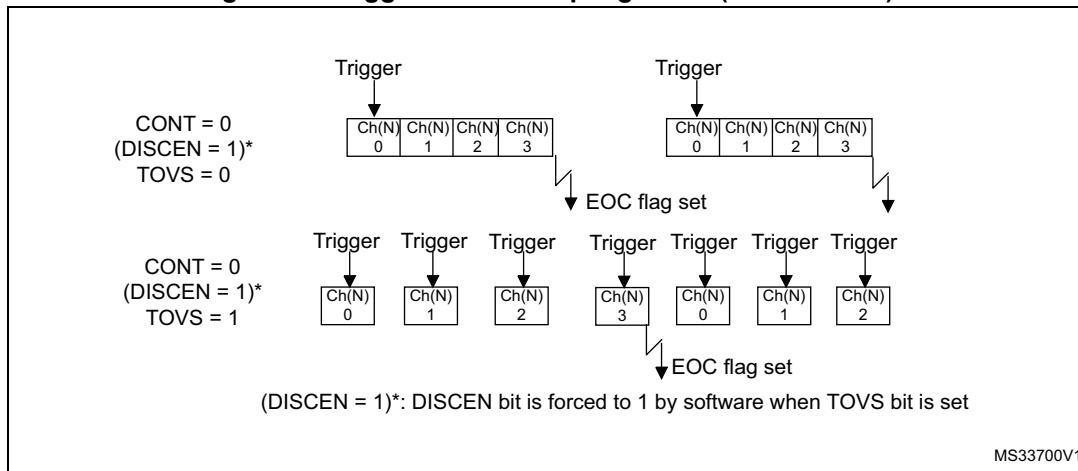
### 14.9.3 Triggered mode

The averager can also be used for basic filtering purposes. Although not a very efficient filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TOVS bit in ADC\_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

*Figure 53* below shows how conversions are started in response to triggers in discontinuous mode.

If the TOVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 53. Triggered oversampling mode (TOVS bit = 1)



## 14.10 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the junction temperature ( $T_J$ ) of the device. The temperature sensor is internally connected to the ADC  $V_{IN}[9]$  input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor analog pin must be greater than the minimum  $T_{S\_temp}$  value specified in the datasheet. When not in use, the sensor can be put in power down mode.

The internal voltage reference ( $V_{REFINT}$ ) provides a stable (bandgap) voltage output for the ADC.  $V_{REFINT}$  is internally connected to the ADC  $V_{IN}[10]$  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.

*Figure 54* shows the block diagram of connections between the temperature sensor, the internal voltage reference and the ADC.

The TSEN bit must be set to enable the conversion of ADC  $V_{IN}[9]$  (temperature sensor) and the VREFEN bit must be set to enable the conversion of ADC  $V_{IN}[10]$  ( $V_{REFINT}$ ).

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

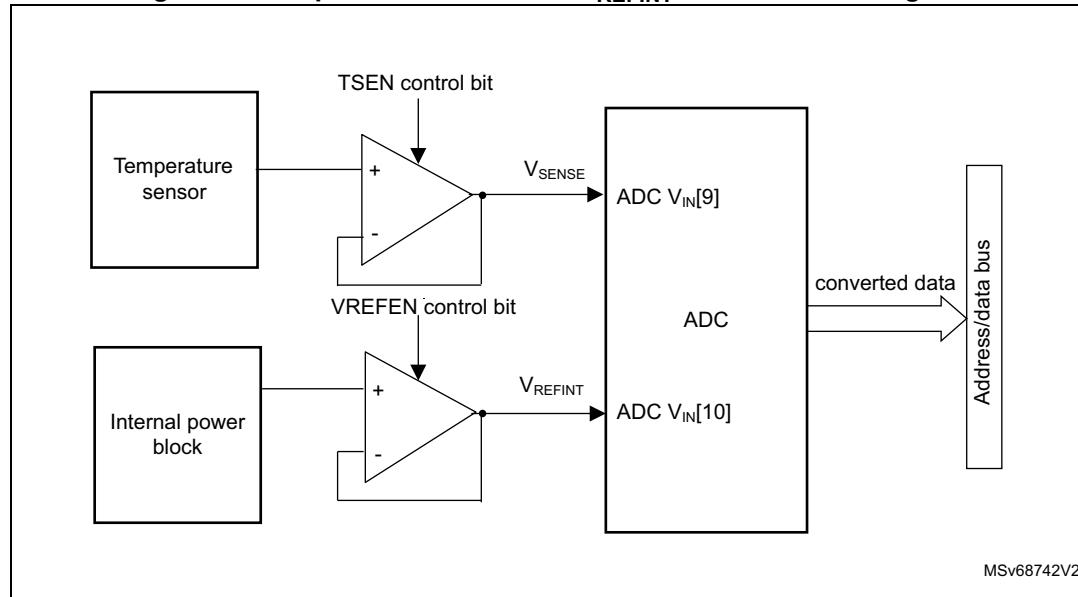
The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the datasheet for additional information.

## Main features

- Linearity:  $\pm 2^\circ\text{C}$  max., precision depending on calibration

**Figure 54. Temperature sensor and  $V_{\text{REFINT}}$  channel block diagram**



## Reading the temperature

1. Select the ADC  $V_{\text{IN}}[9]$  input channel.
2. Select an appropriate sampling time specified in the device datasheet ( $T_{S\_temp}$ ).
3. Set the TSEN bit in the ADC\_CCR register to wake up the temperature sensor from power down mode and wait for its stabilization time ( $t_{\text{START}}$ ).
4. Start the ADC conversion by setting the ADSTART bit in the ADC\_CR register (or by external trigger).
5. Read the resulting  $V_{\text{SENSE}}$  data in the ADC\_DR register.
6. Calculate the temperature using the following formula

$$\text{Temperature (in } ^\circ\text{C)} = \frac{\text{Sense\_DATA} - \text{TS\_CAL1}}{\text{Avg\_Slope\_Code}} + \text{TS\_CAL1\_TEMP}$$

$$\text{Avg\_Slope\_Code} = \text{Avg\_Slope} \times 4096 / 3000$$

$$\text{Sense\_DATA} = \text{TS\_DATA} \times V_{\text{DDA}} / 3.0$$

Where:

- TS\_CAL1 is the temperature sensor calibration value acquired at TS\_CAL1\_TEMP (refer to the datasheet for TS\_CAL1 value)
- TS\_DATA is the actual temperature sensor output value converted by ADC Refer to the specific device datasheet for more information about TS\_CAL1 calibration point.
- Avg\_Slope is the coefficient of the temperature sensor output voltage expressed in mV/°C (refer to the datasheet for Avg\_Slope value).

Note:

*The sensor has a startup time after waking from power down mode before it can output VSENSE at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and TSEN bits should be set at the same time.*

### Calculating the actual V<sub>DDA</sub> voltage using the internal reference voltage

The V<sub>DDA</sub> power supply voltage applied to the device may be subject to variation or not precisely known. The embedded internal voltage reference (V<sub>REFINT</sub>) and its calibration data, acquired by the ADC during the manufacturing process at V<sub>DDA\_Charac</sub>, can be used to evaluate the actual V<sub>DDA</sub> voltage level.

The following formula gives the actual V<sub>DDA</sub> voltage supplying the device:

$$V_{DDA} = V_{DDA\_Charac} \times VREFINT\_CAL / VREFINT\_DATA$$

Where:

- V<sub>DDA\_Charac</sub> is the value of V<sub>DDA</sub> voltage characterized at V<sub>REFINT</sub> during the manufacturing process. It is specified in the device datasheet.
- VREFINT\_CAL is the VREFINT calibration value
- VREFINT\_DATA is the actual VREFINT output value converted by ADC

### Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of V<sub>DDA</sub>. For applications where V<sub>DDA</sub> is known and ADC converted values are right-aligned you can use the following formula to get this absolute value:

$$V_{CHANNELx} = \frac{V_{DDA}}{\text{NUM\_CODES}} \times \text{ADC\_DATA}_x$$

For applications where V<sub>DDA</sub> value is not known, you must use the internal voltage reference and V<sub>DDA</sub> can be replaced by the expression provided in [Section : Calculating the actual V<sub>DDA</sub> voltage using the internal reference voltage](#), resulting in the following formula:

$$V_{CHANNELx} = \frac{V_{DDA\_Charac} \times VREFINT\_CAL \times \text{ADC\_DATA}_x}{VREFINT\_DATA \times \text{NUM\_CODES}}$$

Where:

- $V_{DDA\_Charac}$  is the value of  $V_{DDA}$  voltage characterized at  $V_{REFINT}$  during the manufacturing process. It is specified in the device datasheet.
- $VREFINT\_CAL$  is the VREFINT calibration value
- $ADC\_DATA_x$  is the value measured by the ADC on channelx (right-aligned)
- $VREFINT\_DATA$  is the actual VREFINT output value converted by the ADC
- $NUM\_CODES$  is the number of ADC output codes. For example with 12-bit resolution, it is  $2^{12} = 4096$  or with 8-bit resolution,  $2^8 = 256$ .

**Note:** *If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.*

## 14.11 ADC interrupts

An interrupt can be generated by any of the following events:

- End Of Calibration (EOCAL flag)
- ADC power-up, when the ADC is ready (ADRDY flag)
- End of any conversion (EOC flag)
- End of a sequence of conversions (EOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, AWD3 flags)
- When the Channel configuration is ready (CCRDY flag)
- When the end of sampling phase occurs (EOSMP flag)
- when a data overrun occurs (OVR flag)

Separate interrupt enable bits are available for flexibility.

**Table 60. ADC interrupts**

Interrupt event	Event flag	Enable control bit
End Of Calibration	EOCAL	EOCALIE
ADC ready	ADRDY	ADRDYIE
End of conversion	EOC	EOCIE
End of sequence of conversions	EOS	EOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
Channel Configuration Ready	CCRDY	CCRDYIE
End of sampling phase	EOSMP	EOSMPIE
Overrun	OVR	OVRIE

## 14.12 ADC registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

### 14.12.1 ADC interrupt and status register (ADC\_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

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.	CCRDY	Res.	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
		rc_w1		rc_w1		rc_w1	rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 13 **CCRDY**: Channel Configuration Ready flag

This flag bit is set by hardware when the channel configuration is applied after programming to ADC\_CHSEL register or changing CHSELRMOD or SCANDIR. It is cleared by software by programming it to it.

0: Channel configuration update not applied.

1: Channel configuration update is applied.

*Note: When the software configures the channels (by programming ADC\_CHSEL or changing CHSELRMOD or SCANDIR), it must wait until the CCRDY flag rises before configuring again or starting conversions, otherwise the new configuration (or the START bit) is ignored. Once the flag is asserted, if the software needs to configure again the channels, it must clear the CCRDY flag before proceeding with a new configuration.*

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EOCAL**: End Of Calibration flag

This bit is set by hardware when calibration is complete. It is cleared by software writing 1 to it.

0: Calibration is not complete

1: Calibration is complete

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC\_AWD3TR and ADC\_AWD3TR registers. It is cleared by software by programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC\_AWD2TR and ADC\_AWD2TR registers. It is cleared by software programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

**Bit 7 AWD1:** Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in ADC\_TR1 and ADC\_HR1 registers. It is cleared by software by programming it to 1.

0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog event occurred

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

**Bit 4 OVR:** ADC overrun

This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

**Bit 3 EOS:** End of sequence flag

This bit is set by hardware at the end of the conversion of a sequence of channels selected by the CHSEL bits. It is cleared by software writing 1 to it.

0: Conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Conversion sequence complete

**Bit 2 EOC:** End of conversion flag

This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC\_DR register. It is cleared by software writing 1 to it or by reading the ADC\_DR register.

0: Channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Channel conversion complete

**Bit 1 EOSMP:** End of sampling flag

This bit is set by hardware during the conversion, at the end of the sampling phase. It is cleared by software by programming it to '1'.

0: Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

**Bit 0 ADRDY:** ADC ready

This bit is set by hardware after the ADC has been enabled (ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

### 14.12.2 ADC interrupt enable register (ADC\_IER)

Address offset: 0x04

Reset value: 0x0000 0000

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.	CCRD YIE	Res.	EOCAL IE	Res.	AWD3I E	AWD2I E	AWD1I E	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMP IE	ADRDY IE
		rw		rw		rw	rw	rw			rw	rw	rw	rw	rw

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

Bit 13 **CCRDYIE**: Channel Configuration Ready Interrupt enable

This bit is set and cleared by software to enable/disable the channel configuration ready interrupt.

0: Channel configuration ready interrupt disabled

1: Channel configuration ready interrupt enabled

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bit 11 **EOCALIE**: End of calibration interrupt enable

This bit is set and cleared by software to enable/disable the end of calibration interrupt.

0: End of calibration interrupt disabled

1: End of calibration interrupt enabled

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

*Note:* The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

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

**Bit 4 OVRIE:** Overrun interrupt enable

This bit is set and cleared by software to enable/disable the overrun interrupt.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

*Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).*

**Bit 3 EOSIE:** End of conversion sequence interrupt enable

This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

*Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).*

**Bit 2 EOCIE:** End of conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

0: EOC interrupt disabled

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

*Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).*

**Bit 1 EOSMPIE:** End of sampling flag interrupt enable

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

*Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).*

**Bit 0 ADRDYIE:** ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled.

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

*Note: The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).*

### 14.12.3 ADC control register (ADC\_CR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.							
rs			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.	ADSTP	Res.	ADSTART	ADDIS	ADEN
											rs		rs	rs	rs

#### Bit 31 ADCAL: ADC calibration

This bit is set by software to start the calibration of the ADC.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration is in progress.

*Note: The software is allowed to set ADCAL only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0, AUTOFF = 0, and ADEN = 0).*

*The software is allowed to update the calibration factor by writing ADC\_CALFACT only when ADEN = 1 and ADSTART = 0 (ADC enabled and no conversion is ongoing).*

Bits 30:29 Reserved, must be kept at reset value.

#### Bit 28 ADVREGEN: ADC Voltage Regulator Enable

This bit is set by software, to enable the ADC internal voltage regulator. The voltage regulator output is available after  $t_{ADCVREG\_STUP}$ .

It is cleared by software to disable the voltage regulator. It can be cleared only if ADEN is set to 0.

0: ADC voltage regulator disabled

1: ADC voltage regulator enabled

*Note: The software is allowed to program this bit field only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bits 27:5 Reserved, must be kept at reset value.

#### Bit 4 ADSTP: ADC stop conversion command

This bit is set by software to stop and discard an ongoing conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command.

0: No ADC stop conversion command ongoing

1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.

*Note: Setting ADSTP to '1' is only effective when ADSTART = 1 and ADDIS = 0 (ADC is enabled and may be converting and there is no pending request to disable the ADC)*

Bit 3 Reserved, must be kept at reset value.

**Bit 2 ADSTART: ADC start conversion command**

This bit is set by software to start ADC conversion. Depending on the EXLEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- In single conversion mode (CONT = 0, DISCEN = 0), when software trigger is selected (EXLEN = 00): at the assertion of the end of Conversion Sequence (EOS) flag.
- In discontinuous conversion mode (CONT = 0, DISCEN = 1), when the software trigger is selected (EXLEN = 00): at the assertion of the end of Conversion (EOC) flag.
- In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

0: No ADC conversion is ongoing.

1: Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.

*Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).*

*After writing to ADC\_CHSELR register or changing CHSELRMOD or SCANDIRW, it is mandatory to wait until CCRDY flag is asserted before setting ADSTART, otherwise, the value written to ADSTART is ignored.*

**Bit 1 ADDIS: ADC disable command**

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: No ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

*Note: Setting ADDIS to ‘1’ is only effective when ADEN = 1 and ADSTART = 0 (which ensures that no conversion is ongoing)*

**Bit 0 ADEN: ADC enable command**

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

*Note: The software is allowed to set ADEN only when all bits of ADC\_CR registers are 0 (ADCAL = 0, ADSTP = 0, ADSTART = 0, ADDIS = 0 and ADEN = 0)*

#### 14.12.4 ADC configuration register 1 (ADC\_CFGR1)

Address offset: 0x0C

Reset value: 0x0000 0000

The software is allowed to program ADC\_CFGR1 only when ADEN is cleared in ADC\_CR.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.									AWD1E N	AWD1S GL	CHSEL RMOD	Res.	Res.	Res.	Res.	DISCE N
	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
AUTOF F	WAIT	CONT	OVRM OD	EXTEN[1:0]	Res.				EXTSEL[2:0]	ALIGN	RES[1:0]	SCAND IR	DMAC FG	DMAE N		
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input Channel 0 monitored by AWD

00001: ADC analog input Channel 1 monitored by AWD

.....

10110: ADC analog input Channel 22 monitored by AWD

Others: Reserved

*Note:* The channel selected by the AWDCH[4:0] bits must be also set into the CHSELR register.

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **AWD1EN**: Analog watchdog enable

This bit is set and cleared by software.

0: Analog watchdog 1 disabled

1: Analog watchdog 1 enabled

*Note:*

Bit 22 **AWD1SGL**: Enable the watchdog on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWDCH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

*Note:*

Bit 21 **CHSELRMOD**: Mode selection of the ADC\_CHSELR register

This bit is set and cleared by software to control the ADC\_CHSELR feature:

0: Each bit of the ADC\_CHSELR register enables an input

1: ADC\_CHSELR register is able to sequence up to 8 channels

*Note:* If CCRDY is not yet asserted after channel configuration (writing ADC\_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **DISCEN**: Discontinuous mode

This bit is set and cleared by software to enable/disable discontinuous mode.

0: Discontinuous mode disabled

1: Discontinuous mode enabled

*Note:* It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

Bit 15 **AUTOFF**: Auto-off mode

This bit is set and cleared by software to enable/disable auto-off mode.

0: Auto-off mode disabled

1: Auto-off mode enabled

*Note:*

Bit 14 **WAIT**: Wait conversion mode

This bit is set and cleared by software to enable/disable wait conversion mode.

0: Wait conversion mode off

1: Wait conversion mode on

*Note:*

Bit 13 **CONT**: Single / continuous conversion mode

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

*Note:* It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

Bit 12 **OVRMOD**: Overrun management mode

This bit is set and cleared by software and configure the way data overruns are managed.

0: ADC\_DR register is preserved with the old data when an overrun is detected.

1: ADC\_DR register is overwritten with the last conversion result when an overrun is detected.

*Note:*

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection

These bits are set and cleared by software to select the external trigger polarity and enable the trigger.

00: Hardware trigger detection disabled (conversions can be started by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

*Note:*

Bit 9 Reserved, must be kept at reset value.

**Bits 8:6 EXTSEL[2:0]: External trigger selection**

These bits select the external event used to trigger the start of conversion (refer to [Table 53: External triggers](#) for details):

- 000: TRG0
- 001: TRG1
- 010: TRG2
- 011: TRG3
- 100: TRG4
- 101: TRG5
- 110: TRG6
- 111: TRG7

*Note:*

**Bit 5 ALIGN: Data alignment**

This bit is set and cleared by software to select right or left alignment. Refer to [Figure 41: Data alignment and resolution \(oversampling disabled: OVSE = 0\) on page 259](#)

- 0: Right alignment
- 1: Left alignment

*Note:*

**Bits 4:3 RES[1:0]: Data resolution**

These bits are written by software to select the resolution of the conversion.

- 00: 12 bits
- 01: 10 bits
- 10: 8 bits
- 11: 6 bits

**Bit 2 SCANDIR: Scan sequence direction**

This bit is set and cleared by software to select the direction in which the channels are scanned in the sequence. It is effective only if CHSELMOD bit is cleared.

- 0: Upward scan (from CHSEL0 to CHSEL22)
- 1: Backward scan (from CHSEL22 to CHSEL0)

*Note:*

*If CCRDY is not yet asserted after channel configuration (writing ADC\_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.*

**Bit 1 DMACFG: Direct memory access configuration**

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

- 0: DMA one shot mode selected
- 1: DMA circular mode selected

For more details, refer to [Section 14.6.5: Managing converted data using the DMA on page 261](#)

*Note:*

**Bit 0 DMAEN: Direct memory access enable**

This bit is set and cleared by software to enable the generation of DMA requests. This allows the DMA controller to be used to manage automatically the converted data. For more details, refer to [Section 14.6.5: Managing converted data using the DMA on page 261](#).

- 0: DMA disabled
- 1: DMA enabled

*Note:*

### 14.12.5 ADC configuration register 2 (ADC\_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

The software is allowed to program ADC\_CFGR2 only when ADEN is cleared in ADC\_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKMODE[1:0]	LFTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	Res.	Res.	Res.	TOVS	OVSS[3:0]				OVSR[2:0]			Res.	OVSE
						rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:30 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define how the analog ADC is clocked:

00: ADCCLK (Asynchronous clock mode), generated at product level (refer to RCC section)

01: PCLK/2 (Synchronous clock mode)

10: PCLK/4 (Synchronous clock mode)

11: PCLK (Synchronous clock mode). This configuration must be enabled only if PCLK has a 50% duty clock cycle (APB prescaler configured inside the RCC must be bypassed and the system clock must be 50% duty cycle)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bit 29 **LFTRIG:** Low frequency trigger mode enable

This bit is set and cleared by software.

0: Low Frequency Trigger Mode disabled

1: Low Frequency Trigger Mode enabled

*Note: The software is allowed to write this bit only when ADEN bit is cleared. T*

Bits 28:10 Reserved, must be kept at reset value.

Bit 9 **TOVS:** Triggered Oversampling

This bit is set and cleared by software.

0: All oversampled conversions for a channel are done consecutively after a trigger

1: Each oversampled conversion for a channel needs a trigger

*Note: The software is allowed to write this bit only when ADEN bit is cleared.*

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bit is set and cleared by software.

- 0000: No shift
- 0001: Shift 1-bit
- 0010: Shift 2-bits
- 0011: Shift 3-bits
- 0100: Shift 4-bits
- 0101: Shift 5-bits
- 0110: Shift 6-bits
- 0111: Shift 7-bits
- 1000: Shift 8-bits
- Others: Reserved

*Note: The software is allowed to write this bit only when ADEN bit is cleared.*

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bit filed defines the number of oversampling ratio.

- 000: 2x
- 001: 4x
- 010: 8x
- 011: 16x
- 100: 32x
- 101: 64x
- 110: 128x
- 111: 256x

*Note: The software is allowed to write this bit only when ADEN bit is cleared.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OVSE**: Oversampler Enable

This bit is set and cleared by software.

- 0: Oversampler disabled
- 1: Oversampler enabled

*Note: The software is allowed to write this bit only when ADEN bit is cleared.*

**14.12.6 ADC sampling time register (ADC\_SMPR)**

Address offset: 0x14

Reset value: 0x0000 0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SMPSE L22	SMPSE L21	SMPSE L20	SMPSE L19	SMPSE L18	SMPSE L17	SMPSE L16	SMPSE L15	SMPSE L14	SMPSE L13	SMPSE L12	SMPSE L11	SMPSE L10	SMPSE L9	SMPSE L8	
	rw	rw	rw													
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPSE L7	SMPSE L6	SMPSE L5	SMPSE L4	SMPSE L3	SMPSE L2	SMPSE L1	SMPSE L0	Res.	SMP2[2:0]				Res.	SMP1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:8 **SMPSEL<sub>x</sub>**: Channel-<sub>x</sub> sampling time selection ( $x = 22$  to  $0$ )

These bits are written by software to define which sampling time is used.

0: Sampling time of CHANNEL<sub>x</sub> use the setting of SMP1[2:0] register.

1: Sampling time of CHANNEL<sub>x</sub> use the setting of SMP2[2:0] register.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

*Refer to [Section 14.3: ADC implementation](#) for the maximum number of channels.*

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMP2[2:0]**: Sampling time selection 2

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 160.5 ADC clock cycles

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMP1[2:0]**: Sampling time selection 1

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 160.5 ADC clock cycles

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

### 14.12.7 ADC watchdog threshold register (ADC\_AWD1TR)

Address offset: 0x20

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	HT1[11:0]													
				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.	LT1[11:0]													
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 265](#).

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

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 265](#).

#### 14.12.8 ADC watchdog threshold register (ADC\_AWD2TR)

Address offset: 0x24

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT2[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.	LT2[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 **HT2[11:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 265](#).

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

Bits 11:0 **LT2[11:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 265](#).

### 14.12.9 ADC channel selection register (ADC\_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

- Each ADC\_CHSELR bit enables an input (CHSELRMOD = 0 in ADC\_CFGR1). Refer to the current section.
- ADC\_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC\_CFGR1). Refer to next section.

#### **CHSELRMOD = 0 in ADC\_CFGR1**

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.	CHSEL 22	CHSEL 21	CHSEL 20	CHSEL 19	CHSEL 18	CHSEL 17	CHSEL 16
									rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHSEL 15	CHSEL 14	CHSEL 13	CHSEL 12	CHSEL 11	CHSEL 10	CHSEL 9	CHSEL 8	CHSEL 7	CHSEL 6	CHSEL 5	CHSEL 4	CHSEL 3	CHSEL 2	CHSEL 1	CHSEL 0
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.

#### **Bits 22:0 CHSEL[22:0]: Channel-x selection**

These bits are written by software and define which channels are part of the sequence of channels to be converted. Refer to [Figure 33: ADC connectivity](#) for ADC inputs connected to external channels and internal sources.

0: Input Channel-x is not selected for conversion

1: Input Channel-x is selected for conversion

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

*If CCRDY is not yet asserted after channel configuration (writing ADC\_CHSELR register or changing CHSELRMOD or SCANDIR), the value written to this bit is ignored.*

### 14.12.10 ADC channel selection register [alternate] (ADC\_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

- Each ADC\_CHSELR bit enables an input (CHSELRMOD = 0 in ADC\_CFGR1). Refer to the current previous section.
- ADC\_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC\_CFGR1). Refer to this section.

**CHSELRMOD = 1 in ADC\_CFGR1:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SQ8[3:0]				SQ7[3:0]				SQ6[3:0]				SQ5[3:0]			
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
rw	rw	rw	rw												

Bits 31:28 **SQ8[3:0]:** 8th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates the end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

0000: CH0

0001: CH1

...

1100: CH12

1101: CH13

1110: CH14

1111: No channel selected (End of sequence)

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 27:24 **SQ7[3:0]:** 7th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 23:20 **SQ6[3:0]:** 6th conversion of the sequence

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

**Bits 19:16 SQ5[3:0]: 5th conversion of the sequence**

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

**Bits 15:12 SQ4[3:0]: 4th conversion of the sequence**

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

**Bits 11:8 SQ3[3:0]: 3rd conversion of the sequence**

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

**Bits 7:4 SQ2[3:0]: 2nd conversion of the sequence**

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

**Bits 3:0 SQ1[3:0]: 1st conversion of the sequence**

These bits are programmed by software with the channel number (0...14) assigned to the 8th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

### 14.12.11 ADC watchdog threshold register (ADC\_AWD3TR)

Address offset: 0x2C

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Res.	Res.	Res.	Res.	HT3[11:0]													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Res.	Res.	Res.	Res.	LT3[11:0]													
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT3[11:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 265](#).

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

Bits 11:0 **LT3[11:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to [Section 14.8: Analog window watchdogs on page 265](#).

### 14.12.12 ADC data register (ADC\_DR)

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.	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
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **DATA[15:0]**: Converted data

These bits are read-only. They contain the conversion result from the last converted channel. The data are left- or right-aligned as shown in [Figure 41: Data alignment and resolution \(oversampling disabled: OVSE = 0\) on page 259](#).

Just after a calibration is complete, DATA[6:0] contains the calibration factor.

### 14.12.13 ADC analog watchdog 2 configuration register (ADC\_AWD2CR)

Address offset: 0xA0

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.	AWD2 CH22	AWD2 CH21	AWD2 CH20	AWD2 CH19	AWD2 CH18	AWD2 CH17	AWD2 CH16
									rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2 CH15	AWD2 CH14	AWD2 CH13	AWD2 CH12	AWD2 CH11	AWD2 CH10	AWD2 CH9	AWD2 CH8	AWD2 CH7	AWD2 CH6	AWD2 CH5	AWD2 CH4	AWD2 CH3	AWD2 CH2	AWD2 CH1	AWD2 CH0
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.

Bits 22:0 **AWD2CH[22:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 2 (AWD2).

0: ADC analog channel-x is not monitored by AWD2

1: ADC analog channel-x is monitored by AWD2

*Note:* The channels selected through ADC\_AWD2CR must be also configured into the ADC\_CHSELR registers. The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

### 14.12.14 ADC Analog Watchdog 3 Configuration register (ADC\_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

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.	AWD3 CH22	AWD3 CH21	AWD3 CH20	AWD3 CH19	AWD3 CH18	AWD3 CH17	AWD3 CH16
									rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3 CH15	AWD3 CH14	AWD3 CH13	AWD3 CH12	AWD3 CH11	AWD3 CH10	AWD3 CH9	AWD3 CH8	AWD3 CH7	AWD3 CH6	AWD3 CH5	AWD3 CH4	AWD3 CH3	AWD3 CH2	AWD3 CH1	AWD3 CH0
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.

Bits 22:0 **AWD3CH[22:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 3 (AWD3).

0: ADC analog channel-x is not monitored by AWD3

1: ADC analog channel-x is monitored by AWD3

*Note:* The channels selected through ADC\_AWD3CR must be also configured into the ADC\_CHSELR registers. The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

### 14.12.15 ADC calibration factor (ADC\_CALFACT)

Address offset: 0xB4

Reset value: 0x0000 0000

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.	Res.	Res.	Res.	Res.	Res.	rw	rw	rw	rw	rw	rw
CALFACT[6:0]															

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

Bits 6:0 **CALFACT[6:0]**: Calibration factor

These bits are written by hardware or by software.

- Once a calibration is complete, they are updated by hardware with the calibration factors.
- Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new conversion is launched.
- Just after a calibration is complete, DATA[6:0] contains the calibration factor.

*Note: Software can write these bits only when ADEN=1 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).*

### 14.12.16 ADC common configuration register (ADC\_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSEN	VREF EN	PRESC[3:0]				Res.	Res.							
								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.									

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

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TSEN**: Temperature sensor enable

This bit is set and cleared by software to enable/disable the temperature sensor.

0: Temperature sensor disabled

1: Temperature sensor enabled

*Note: Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 22 **VREFEN**: V<sub>REFINT</sub> enable

This bit is set and cleared by software to enable/disable the V<sub>REFINT</sub>.

- 0: V<sub>REFINT</sub> disabled
- 1: V<sub>REFINT</sub> enabled

*Note:* Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 21:18 **PRESCL[3:0]**: ADC prescaler

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

- 0000: input ADC clock not divided
- 0001: input ADC clock divided by 2
- 0010: input ADC clock divided by 4
- 0011: input ADC clock divided by 6
- 0100: input ADC clock divided by 8
- 0101: input ADC clock divided by 10
- 0110: input ADC clock divided by 12
- 0111: input ADC clock divided by 16
- 1000: input ADC clock divided by 32
- 1001: input ADC clock divided by 64
- 1010: input ADC clock divided by 128
- 1011: input ADC clock divided by 256
- Other: Reserved

*Note:* Software is allowed to write these bits only when the ADC is disabled (ADCAL = 0,

ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 17:0 Reserved, must be kept at reset value.

## 14.13 ADC register map

The following table summarizes the ADC registers.

Table 61. ADC 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
0x00	ADC_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.	
	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	
0x04	ADC_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.	
	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	
0x08	ADC_CR	ADCAL	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	ADC_CFGR1	Res.	AWDCH[4:0]				AWD1EN	AWD1SGL	AWD1RMOD	AWD2EN	AWD2SGL	AWD2RMOD	CONT	CCRDYIE	EOCAL	EOCAL	AWD3IE	AWD3IE	AWD3IE	AWD3IE								
	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	

**Table 61. ADC register map and reset values (continued)**

Refer to [Section 2.2](#) for the register boundary addresses.

## 15 Advanced-control timer (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

### 15.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

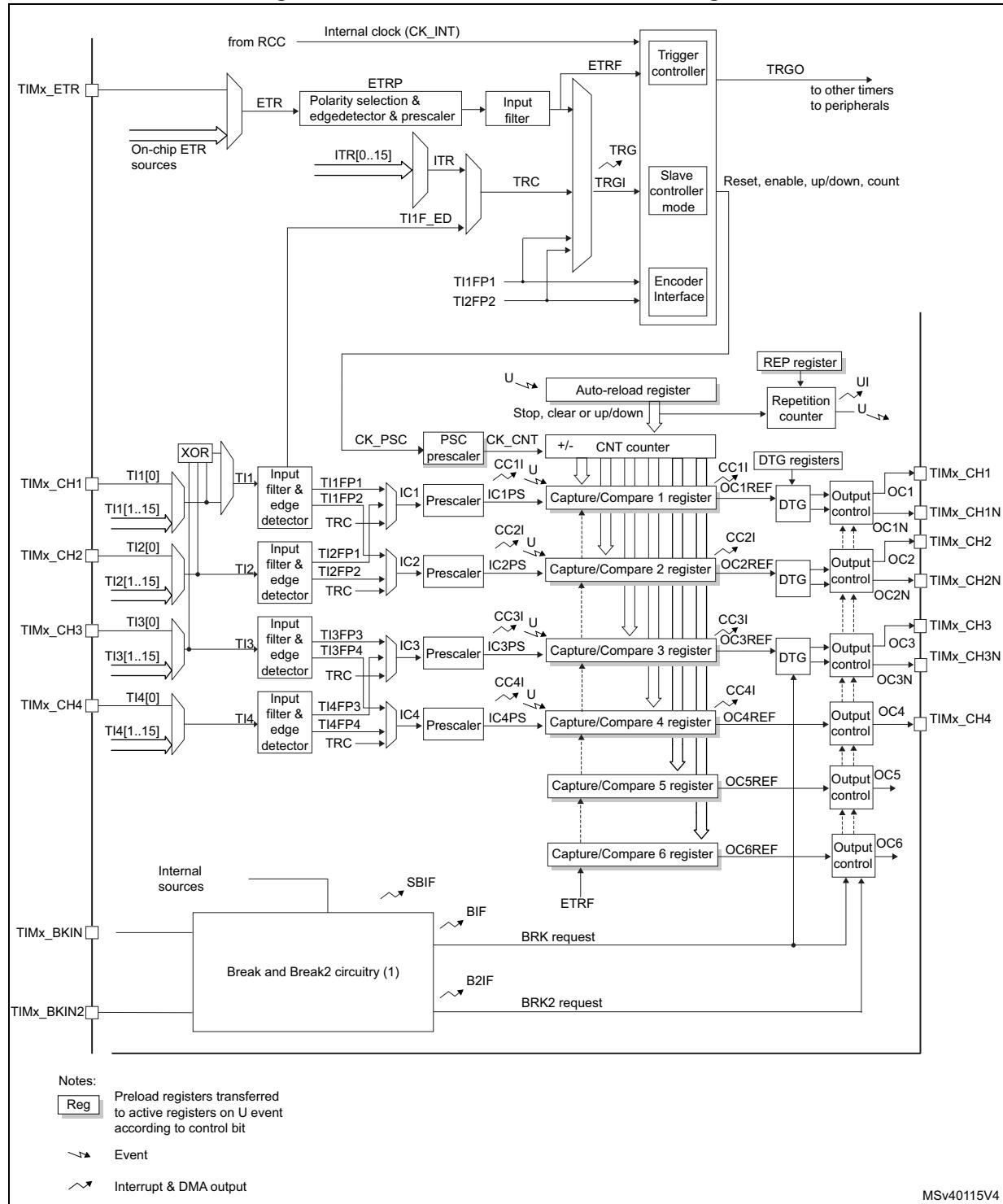
The advanced-control (TIM1) and general-purpose (TMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 15.3.26: Timer synchronization](#).

## 15.2 TIM1 main features

TIM1 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 55. Advanced-control timer block diagram



1. The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 5.2.6: Clock security system \(CSS\)](#)
  - SRAM parity error signal
  - Cortex®-M0+ LOCKUP (Hardfault) output.

## 15.3 TIM1 functional description

### 15.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

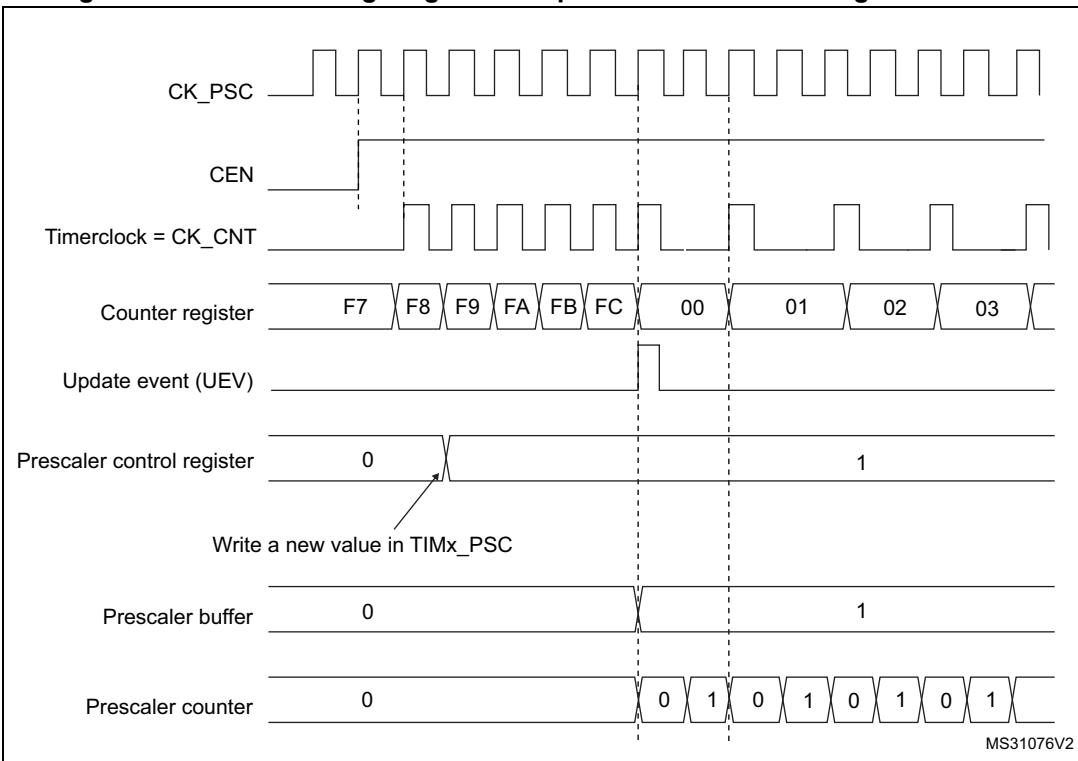
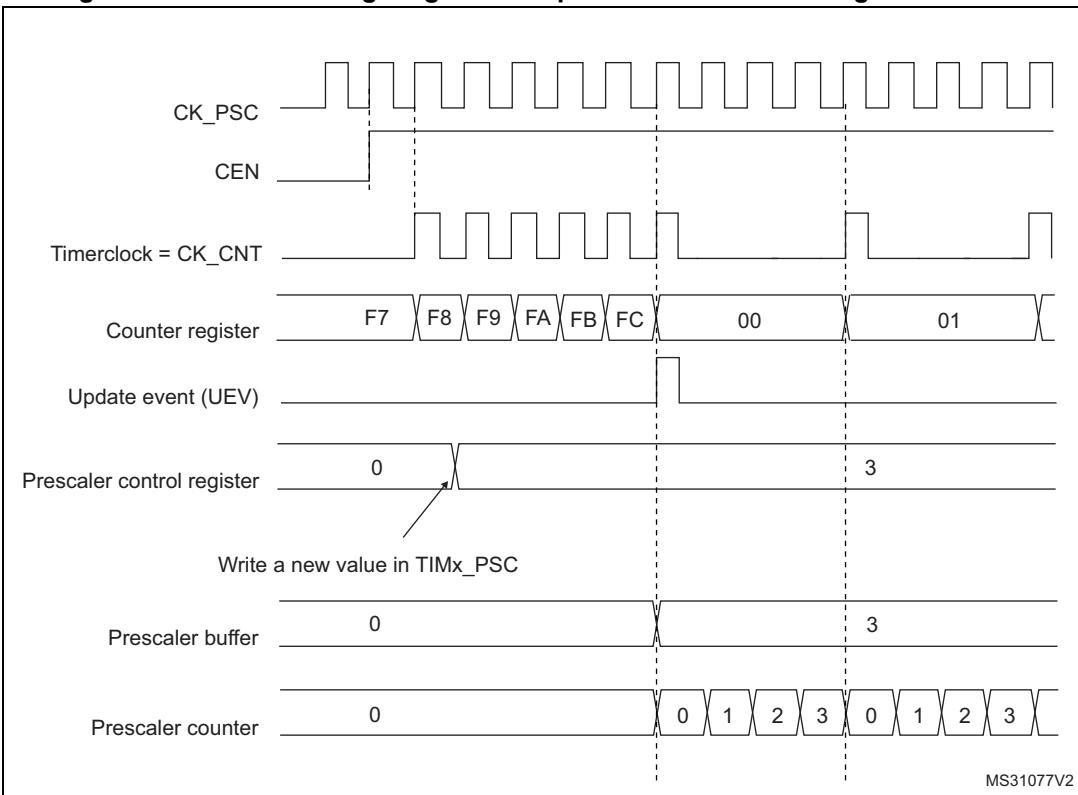
The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 56* and *Figure 57* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 56. Counter timing diagram with prescaler division change from 1 to 2****Figure 57. Counter timing diagram with prescaler division change from 1 to 4**

### 15.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR) + 1. Else the update event is generated at each counter overflow.

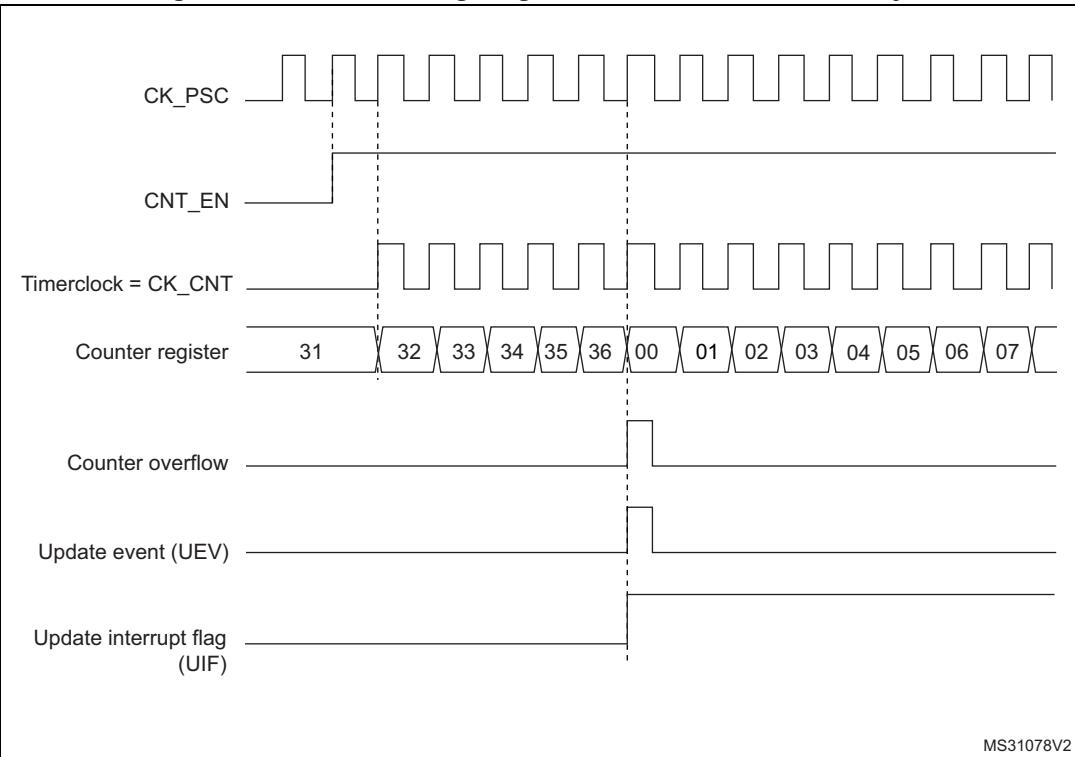
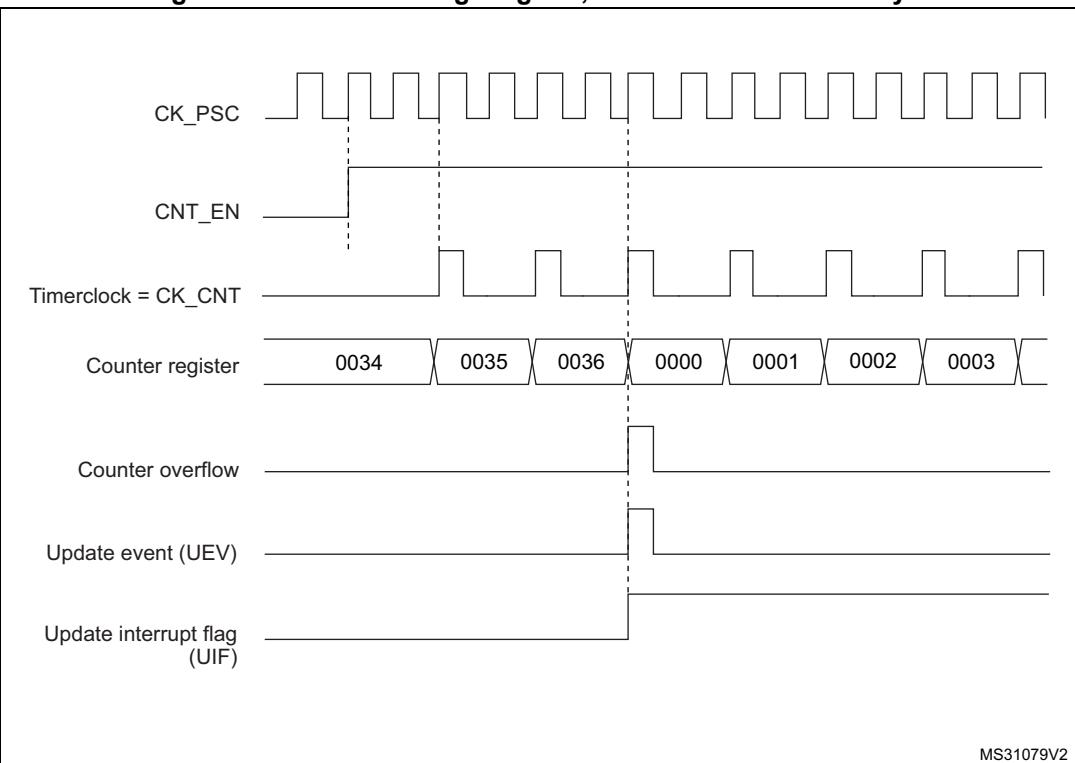
Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

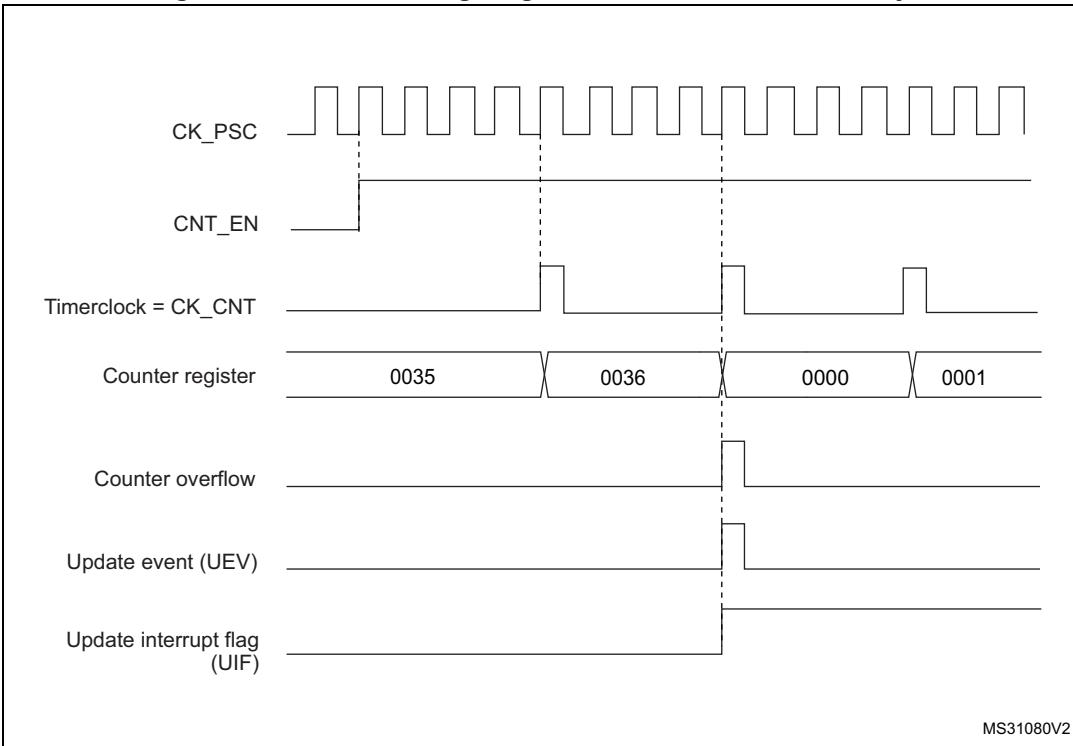
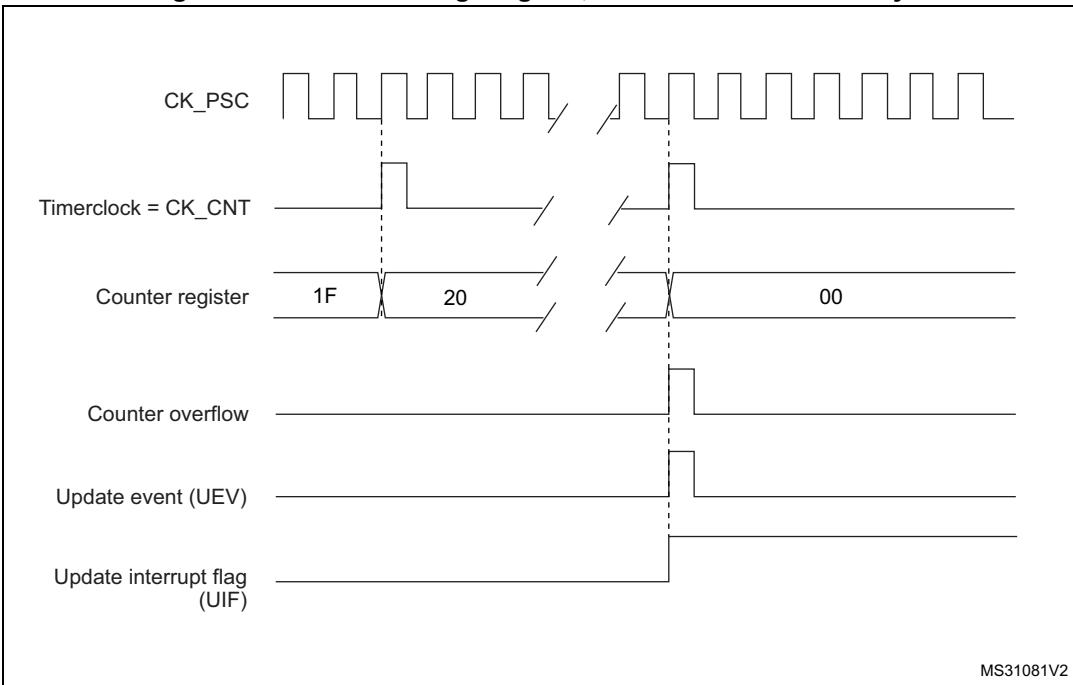
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

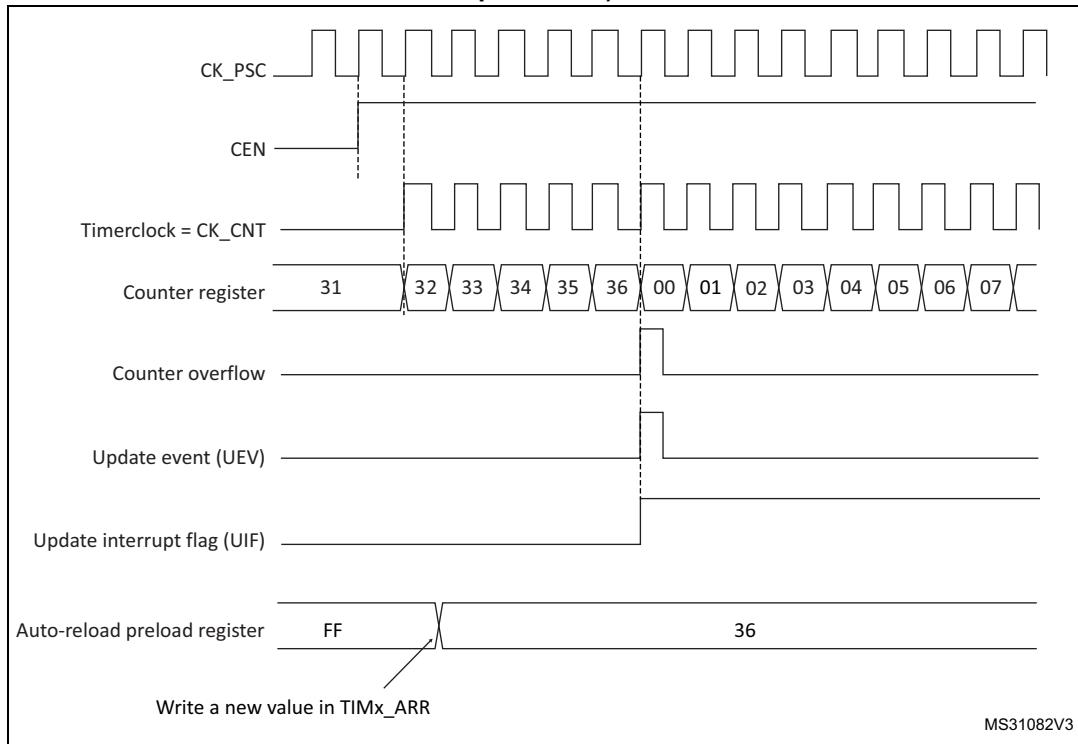
- The repetition counter is reloaded with the content of TIMx\_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

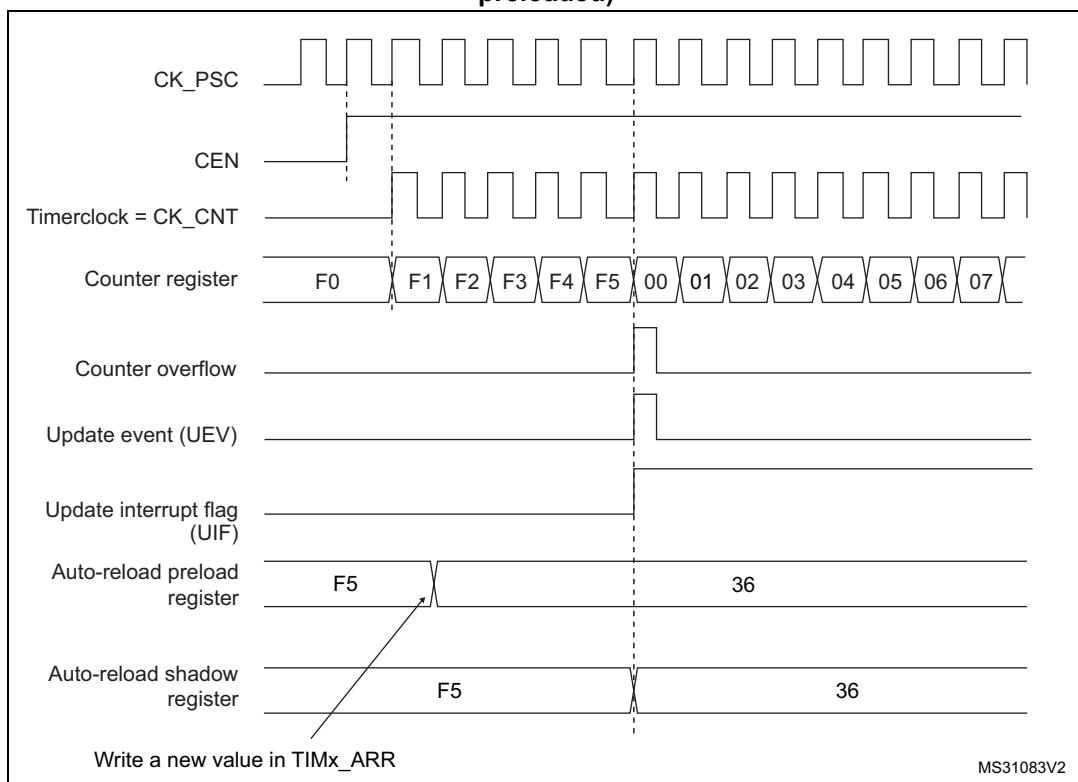
**Figure 58. Counter timing diagram, internal clock divided by 1****Figure 59. Counter timing diagram, internal clock divided by 2**

**Figure 60. Counter timing diagram, internal clock divided by 4****Figure 61. Counter timing diagram, internal clock divided by N**

**Figure 62. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 63. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



## Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx\_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

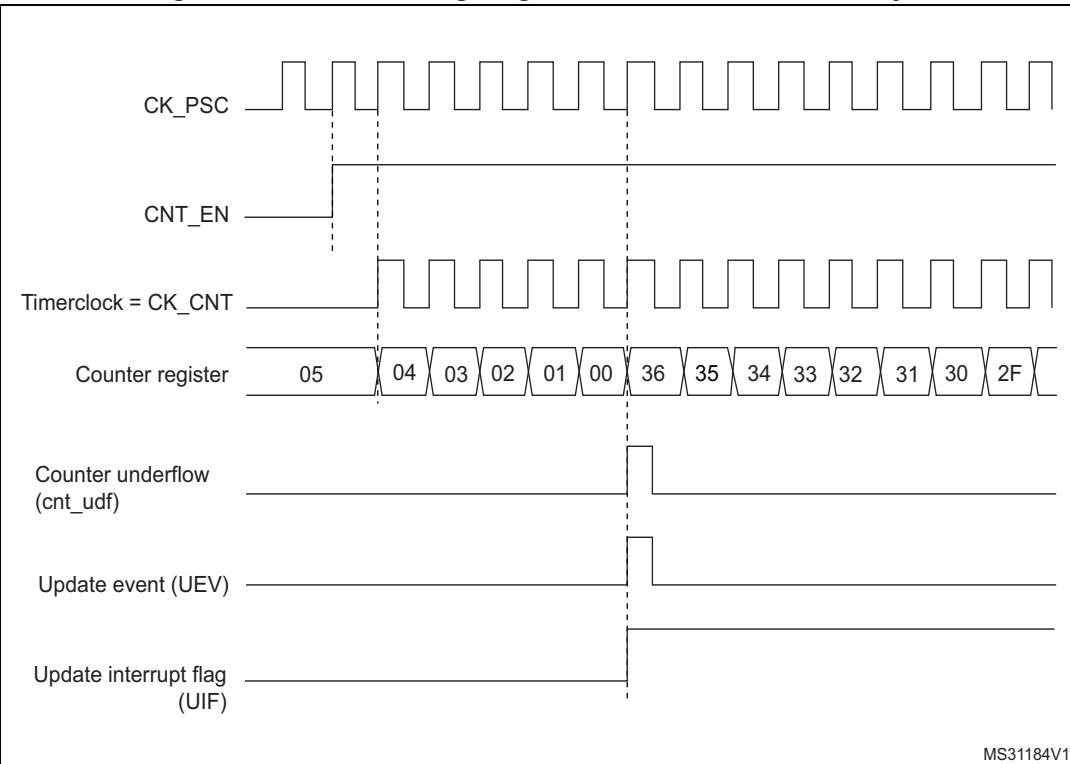
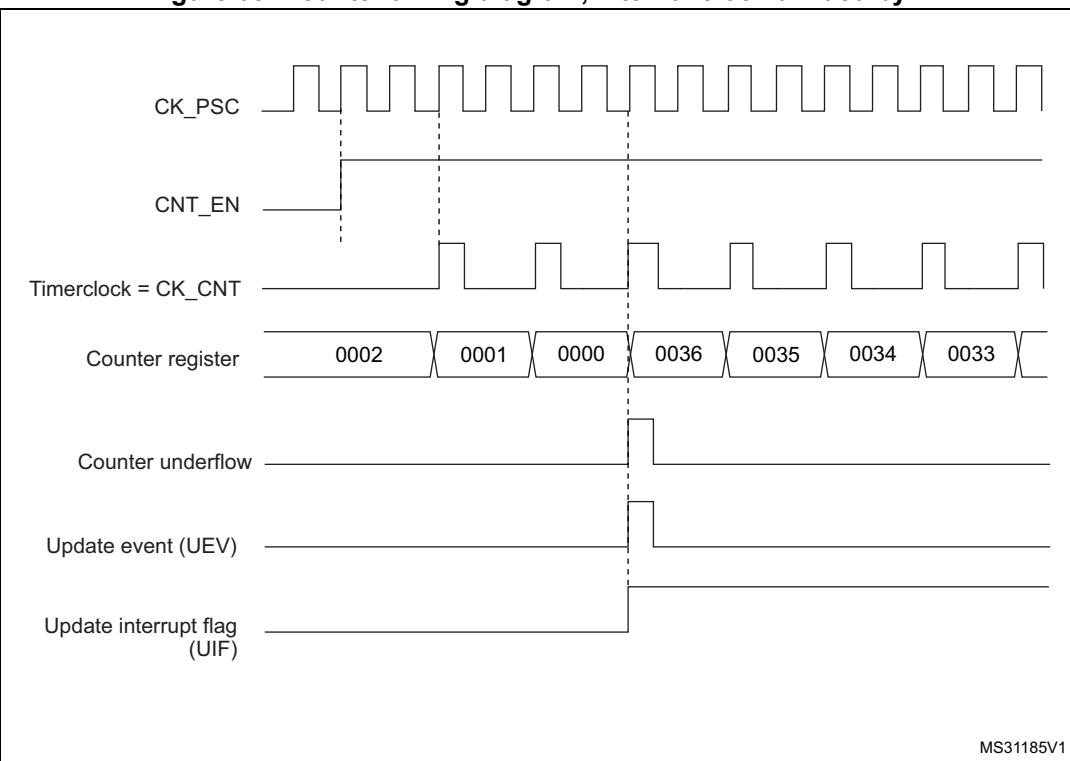
The UEV update event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

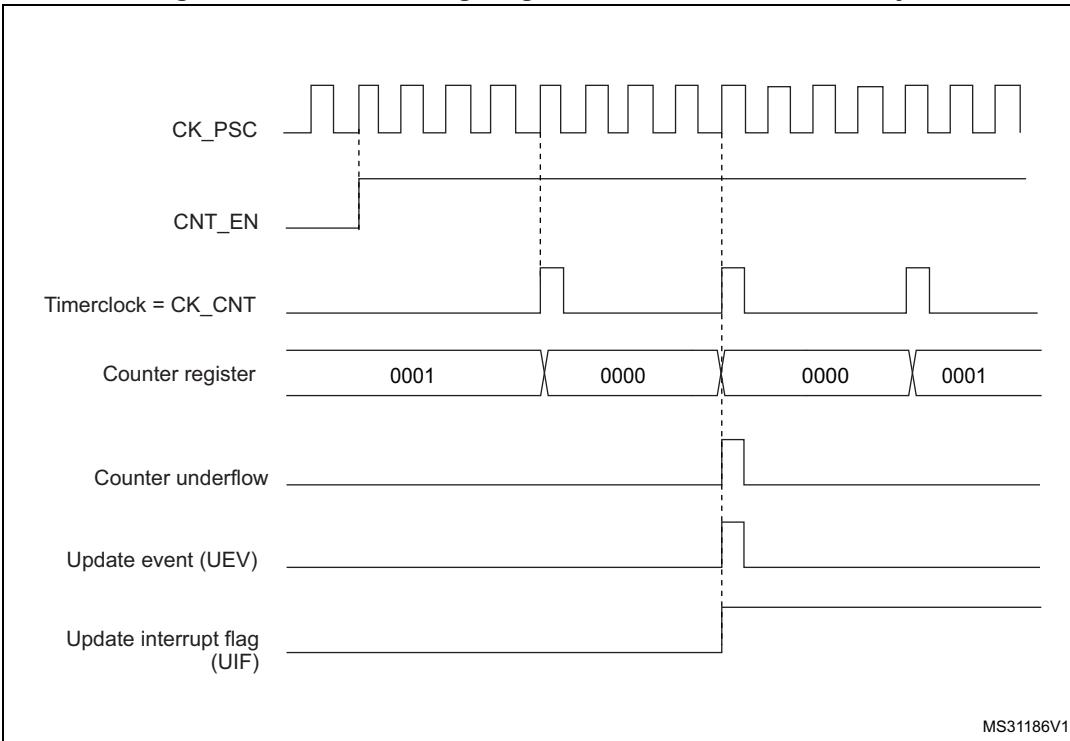
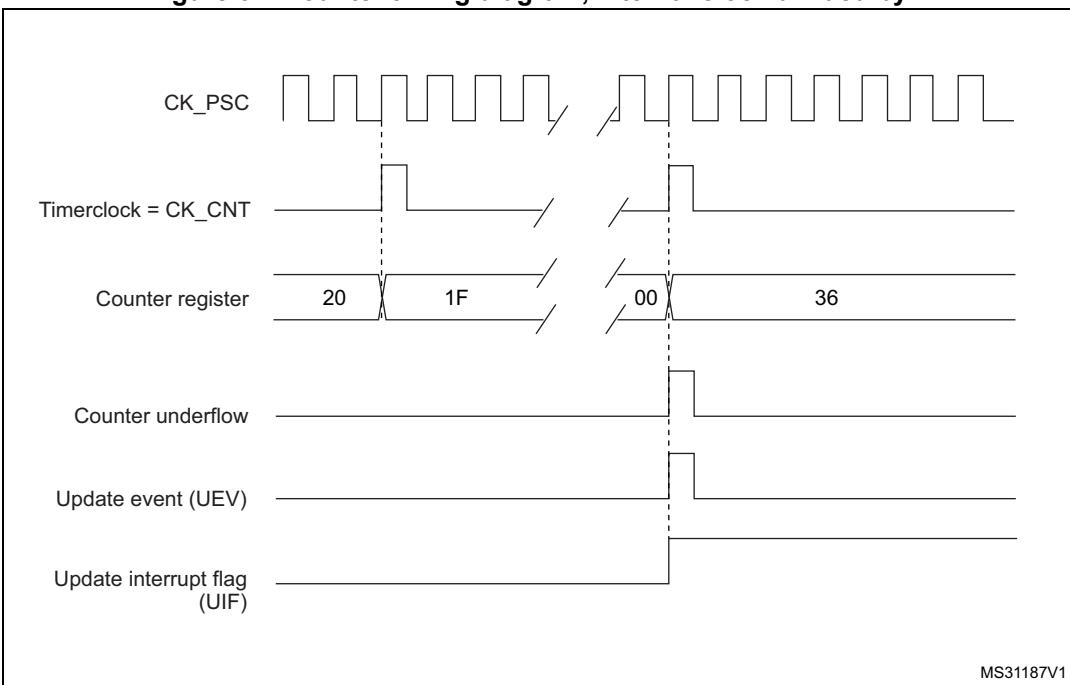
In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

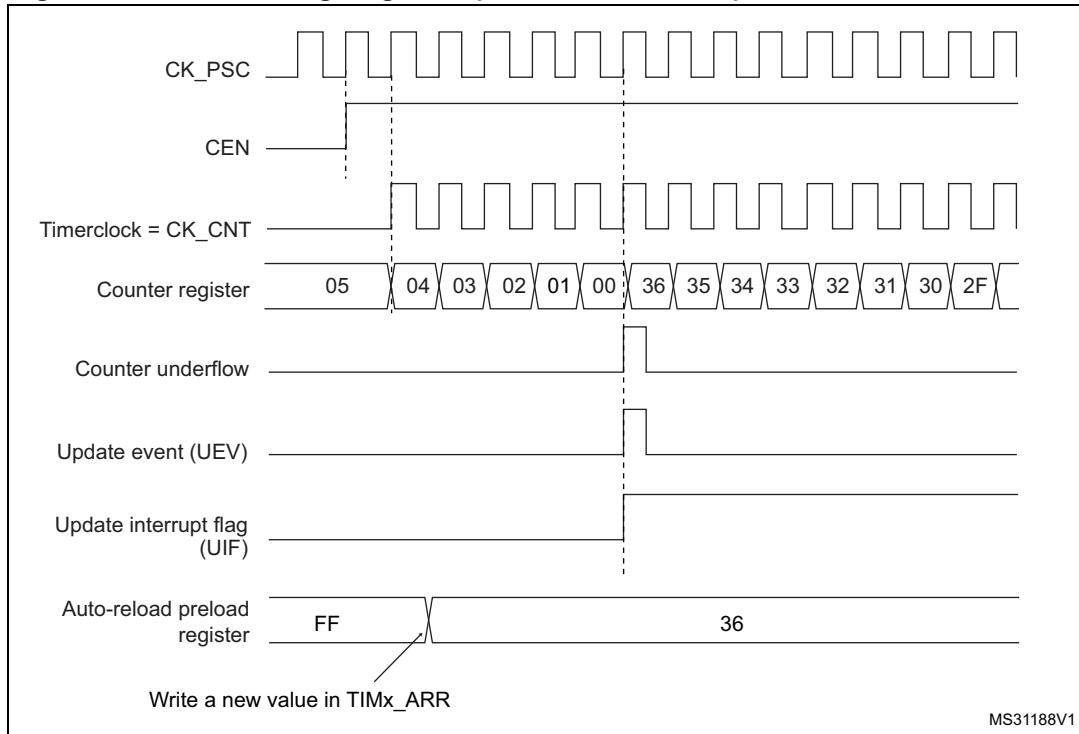
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx\_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 64. Counter timing diagram, internal clock divided by 1****Figure 65. Counter timing diagram, internal clock divided by 2**

**Figure 66. Counter timing diagram, internal clock divided by 4****Figure 67. Counter timing diagram, internal clock divided by N**

**Figure 68. 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") or the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx\_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or

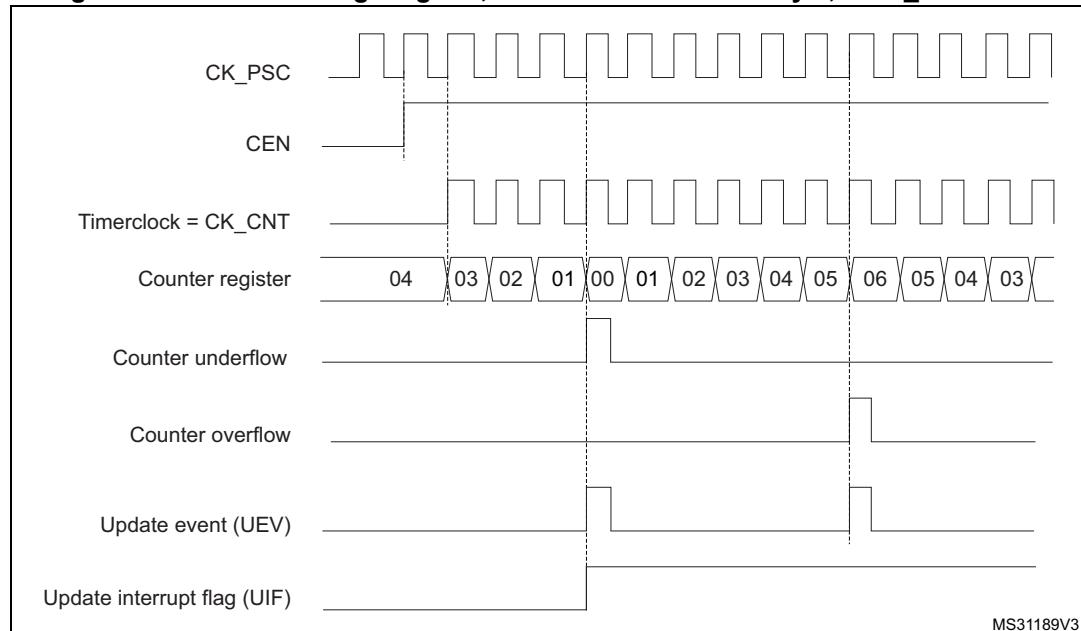
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

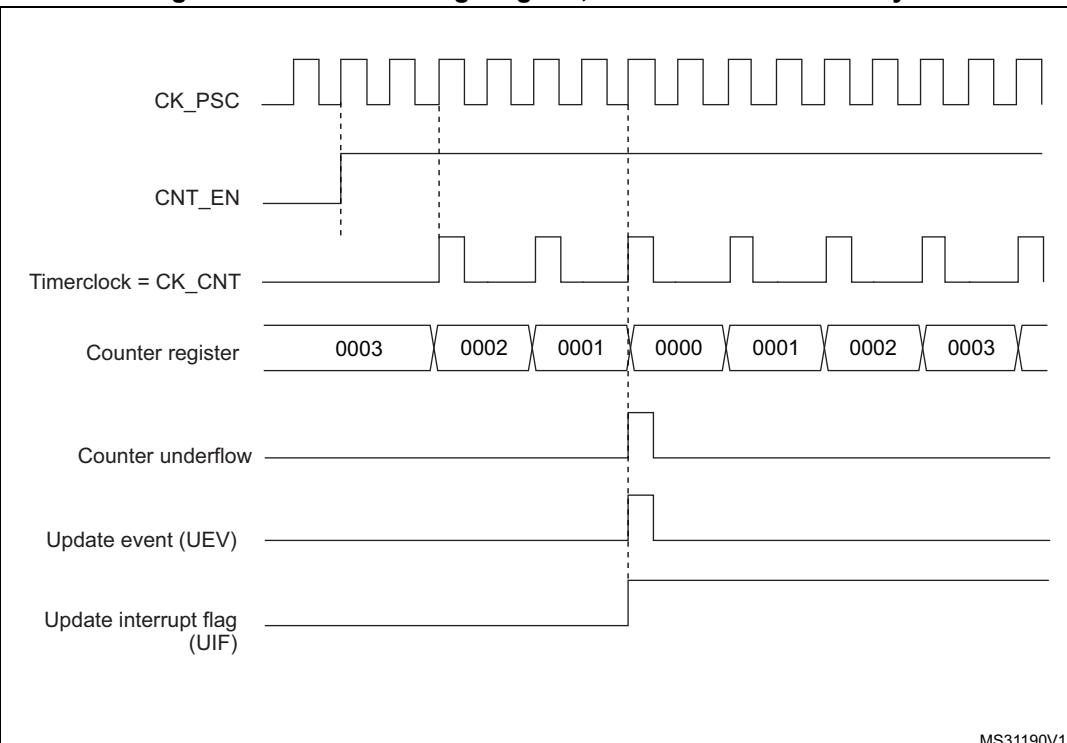
- The repetition counter is reloaded with the content of TIMx\_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

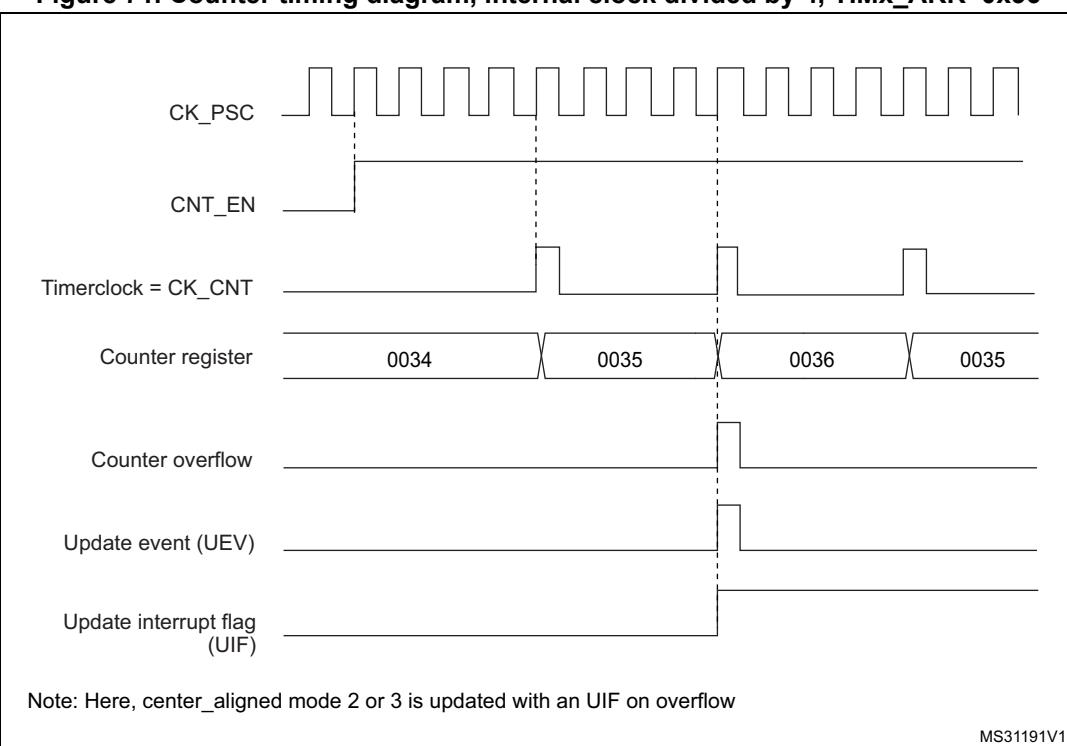
**Figure 69. Counter timing diagram, internal clock divided by 1, TIMx\_ARR = 0x6**



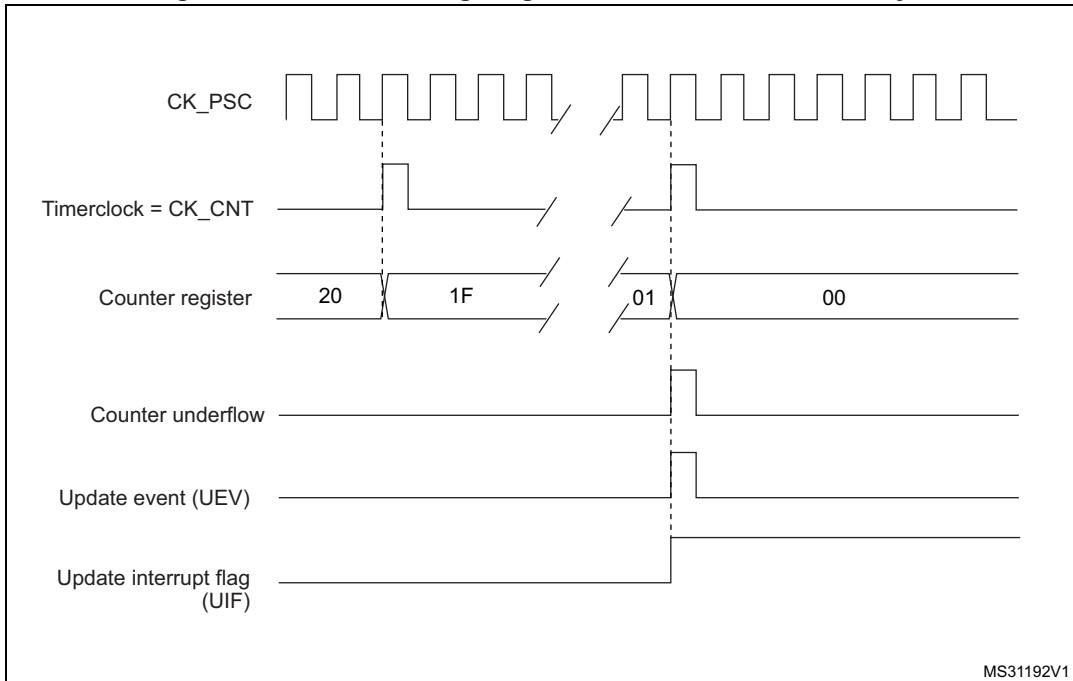
1. Here, center-aligned mode 1 is used (for more details refer to [Section 15.4: TIM1 registers](#)).

**Figure 70. Counter timing diagram, internal clock divided by 2**

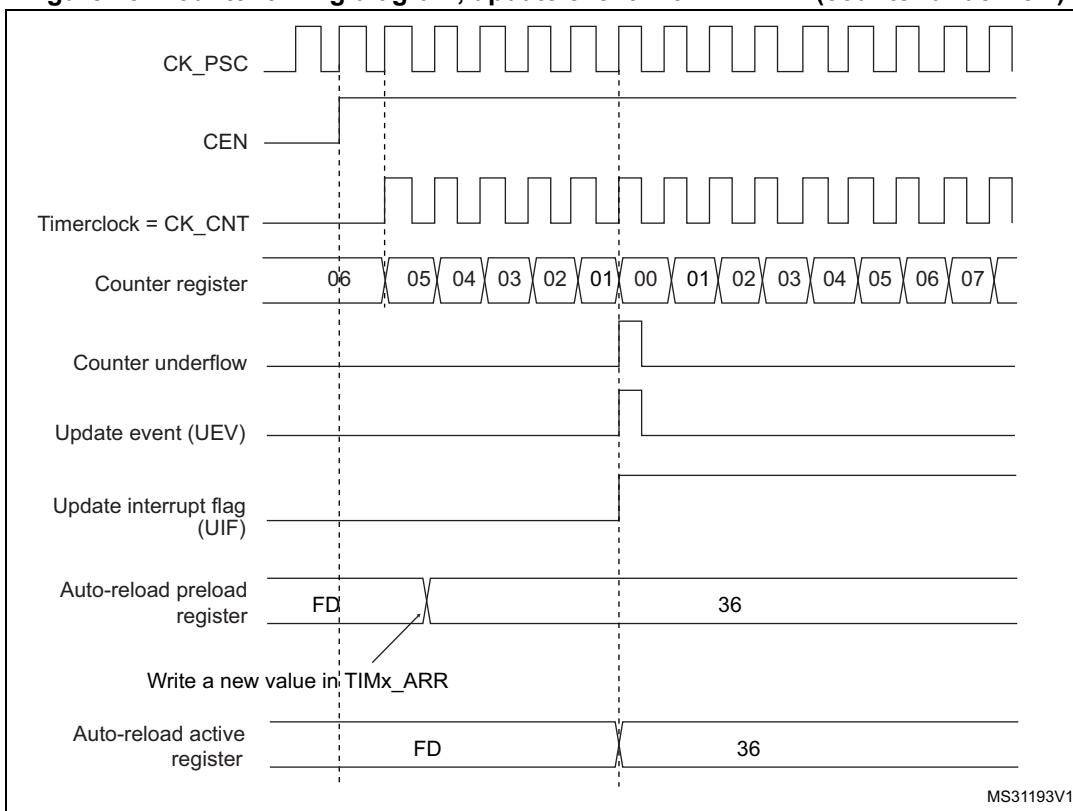
MS31190V1

**Figure 71. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36**

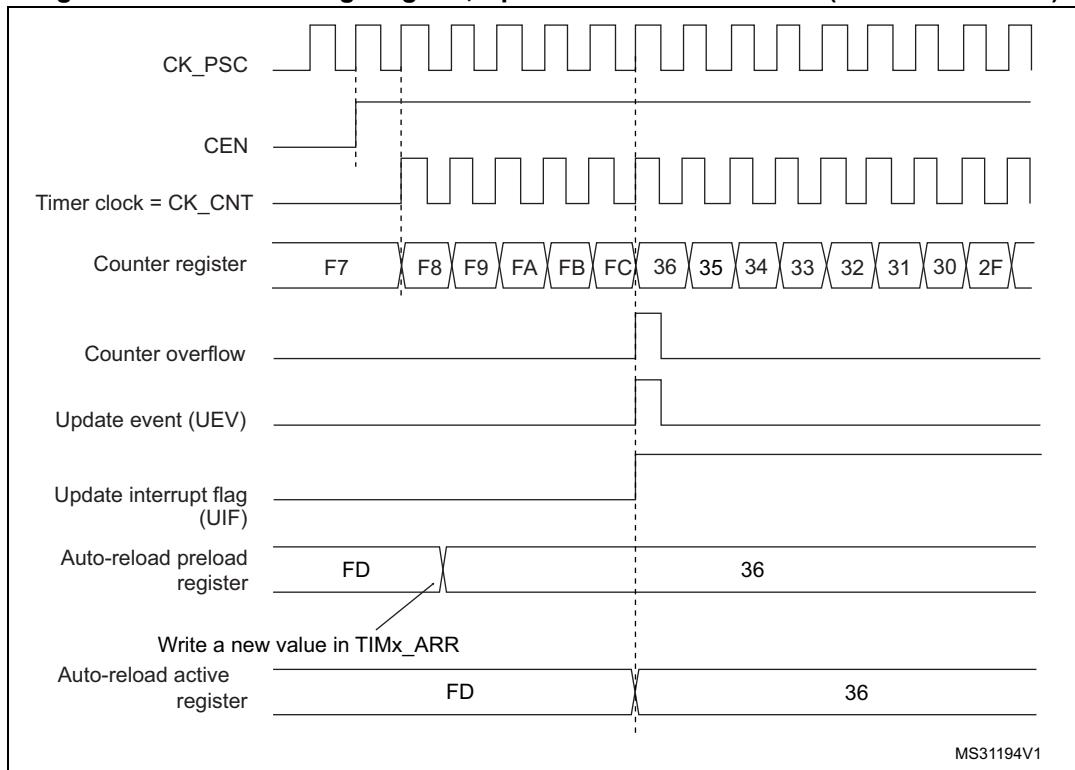
MS31191V1

**Figure 72. Counter timing diagram, internal clock divided by N**

MS31192V1

**Figure 73. Counter timing diagram, update event with ARPE=1 (counter underflow)**

MS31193V1

**Figure 74. Counter timing diagram, Update event with ARPE=1 (counter overflow)**

### 15.3.3 Repetition counter

[Section 15.3.1: Time-base unit](#) describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC prescaler register, but also TIMx\_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented:

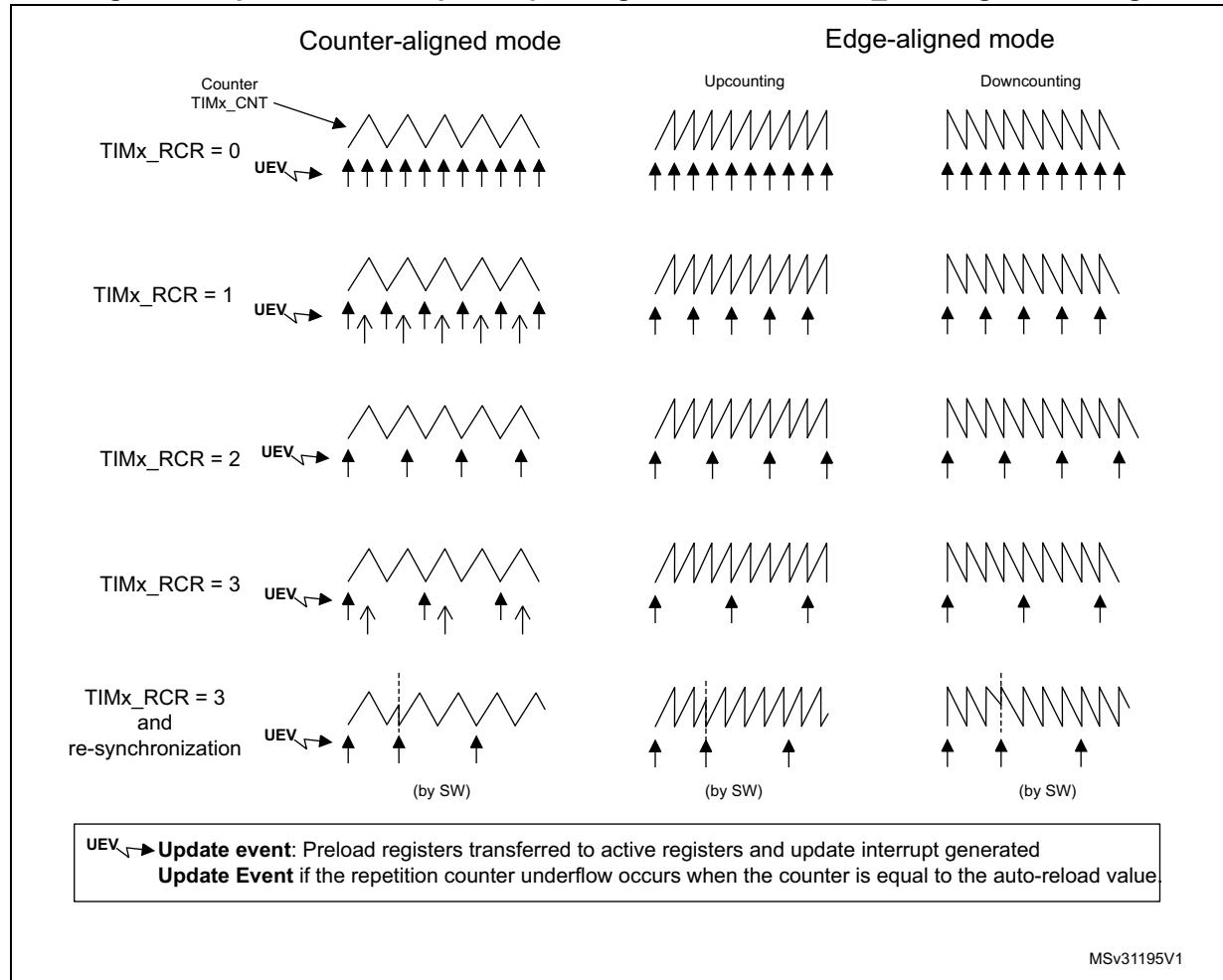
- At each counter overflow in upcounting mode,
  - At each counter underflow in downcounting mode,
  - At each counter overflow and at each counter underflow in center-aligned mode.
- Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is  $2 \times T_{ck}$ , due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to [Figure 75](#)). When the update event is generated by software (by setting the UG bit in TIMx\_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx\_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

**Figure 75. Update rate examples depending on mode and TIMx\_RCR register settings**



MSv31195V1

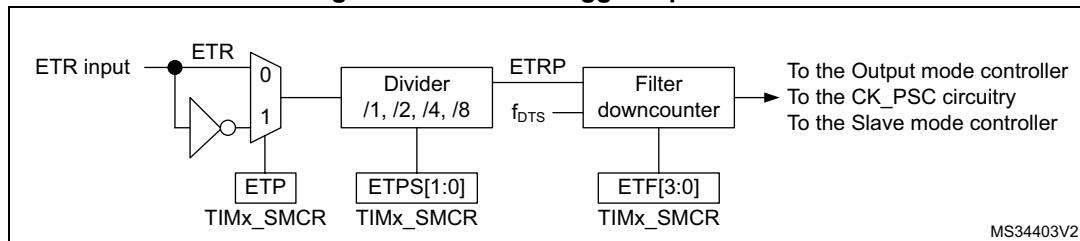
### 15.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see [Section 15.3.5](#))
- trigger for the slave mode (see [Section 15.3.26](#))
- PWM reset input for cycle-by-cycle current regulation (see [Section 15.3.7](#))

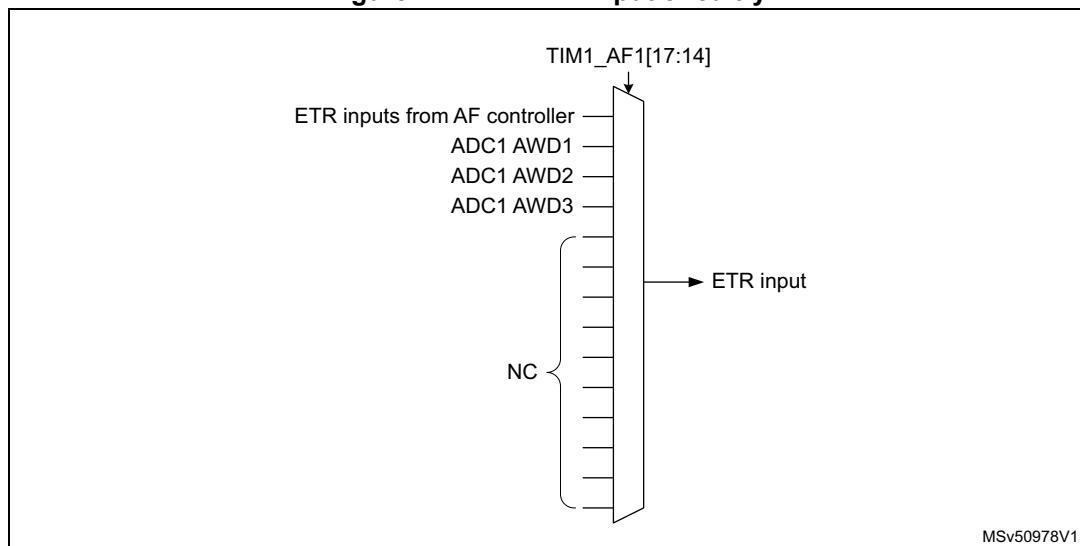
[Figure 76](#) below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

**Figure 76. External trigger input block**



The ETR input comes from multiple sources: input pins (default configuration) and analog watchdogs. The selection is done with the ETRSEL[3:0] bitfield.

**Figure 77. TIM1 ETR input circuitry**



### 15.3.5 Clock selection

The counter clock can be provided by the following clock sources:

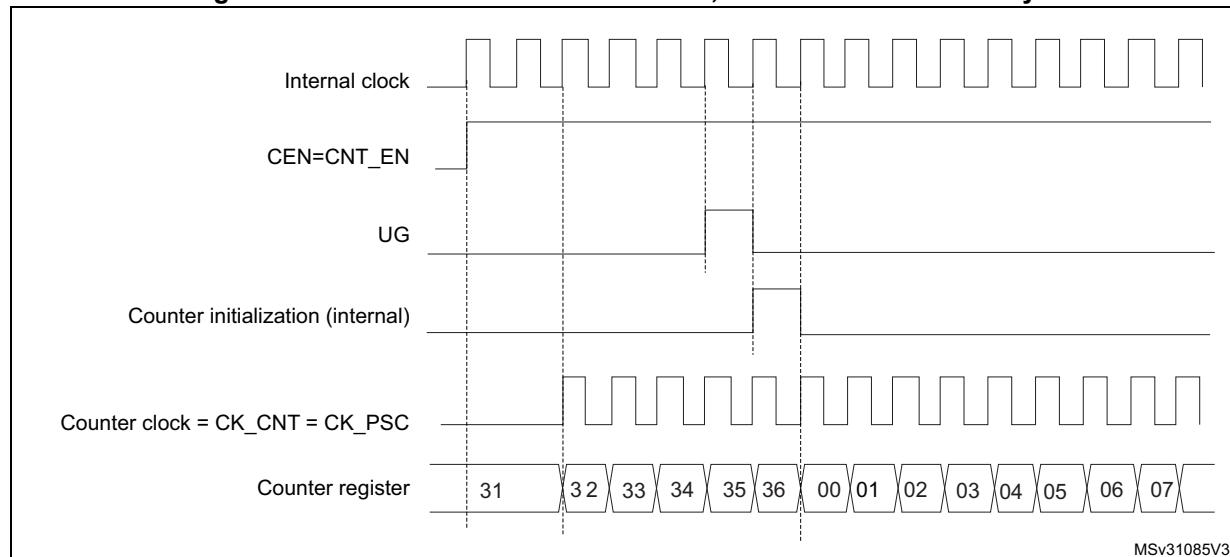
- Internal clock (CK\_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

#### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

*Figure 78* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

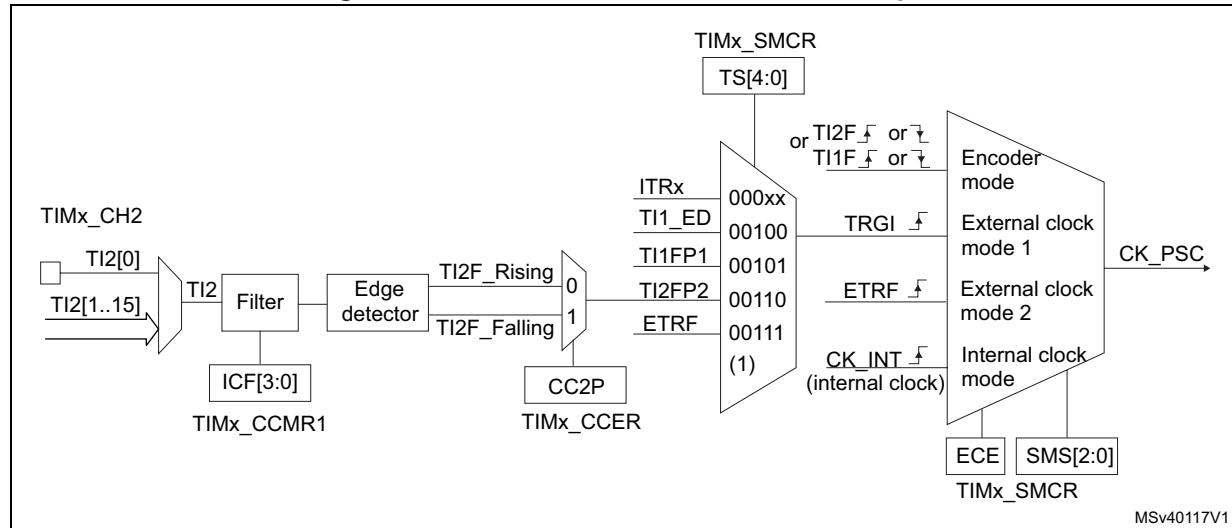
**Figure 78. Control circuit in normal mode, internal clock divided by 1**



#### 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 79. TI2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

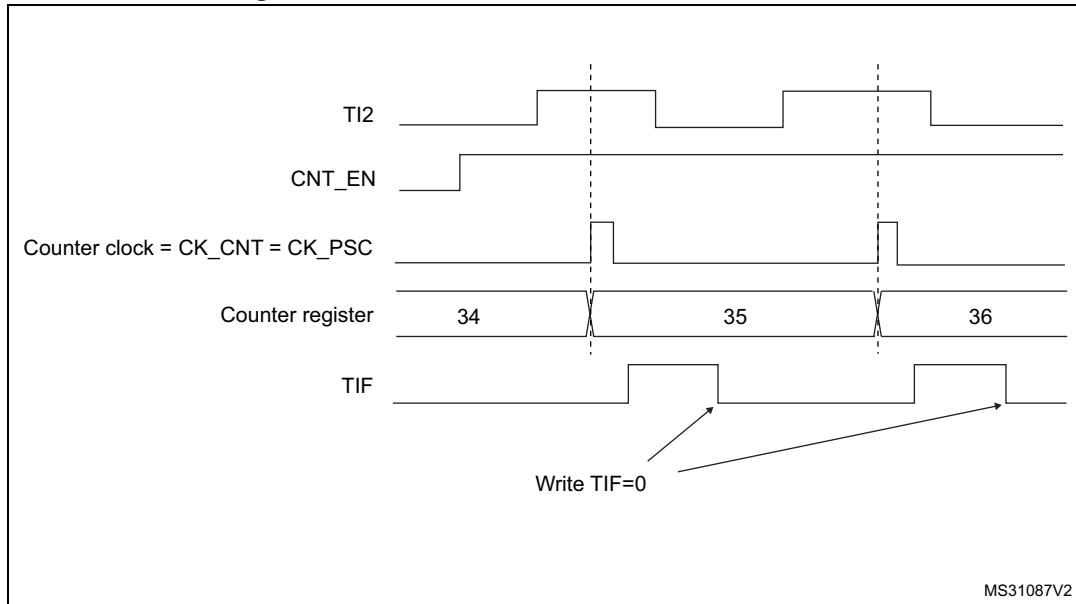
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx\_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx\_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

*Note:*

*The capture prescaler is not used for triggering, so the user does not need to configure it.*

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

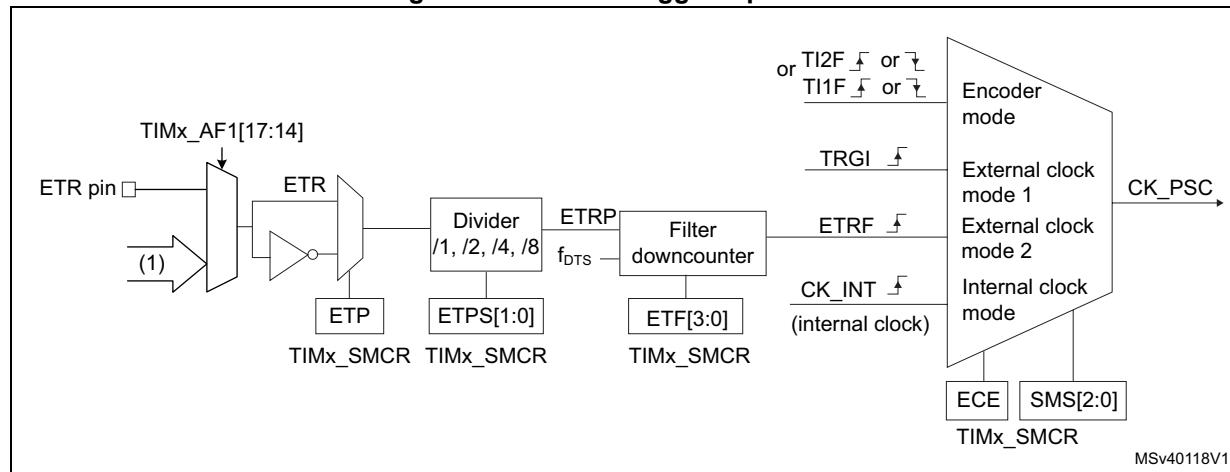
The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 80. Control circuit in external clock mode 1****External clock source mode 2**

This mode is selected by writing ECE=1 in the TIMx\_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The [Figure 81](#) gives an overview of the external trigger input block.

**Figure 81. External trigger input block**

- Refer to [Figure 77: TIM1 ETR input circuitry](#).

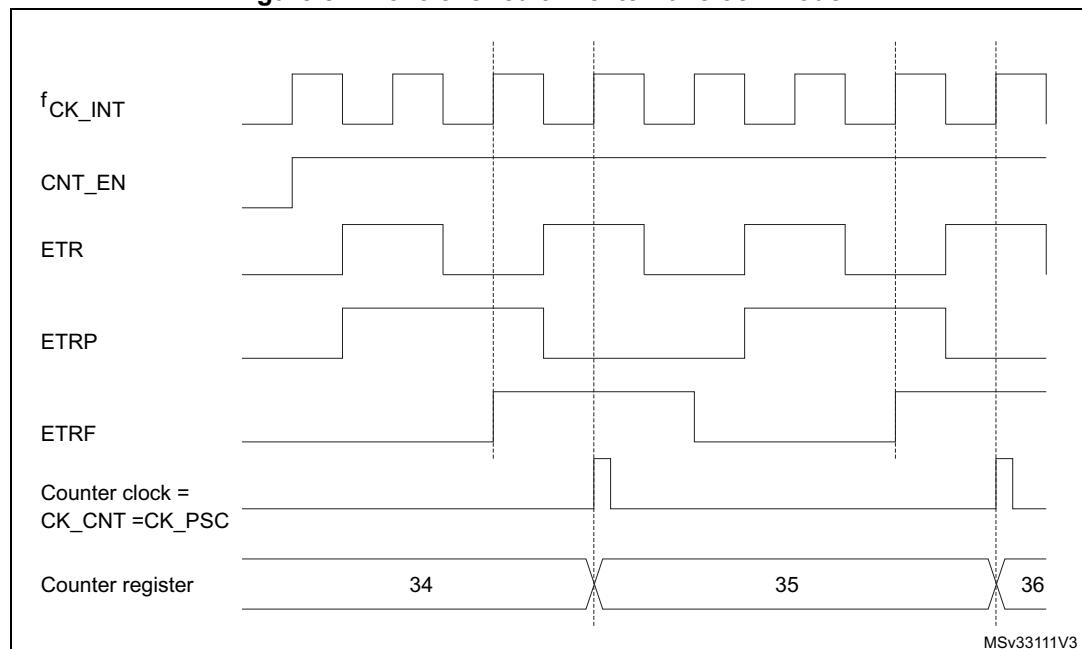
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx\_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx\_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx\_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx\_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most  $\frac{1}{4}$  of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

**Figure 82. Control circuit in external clock mode 2**



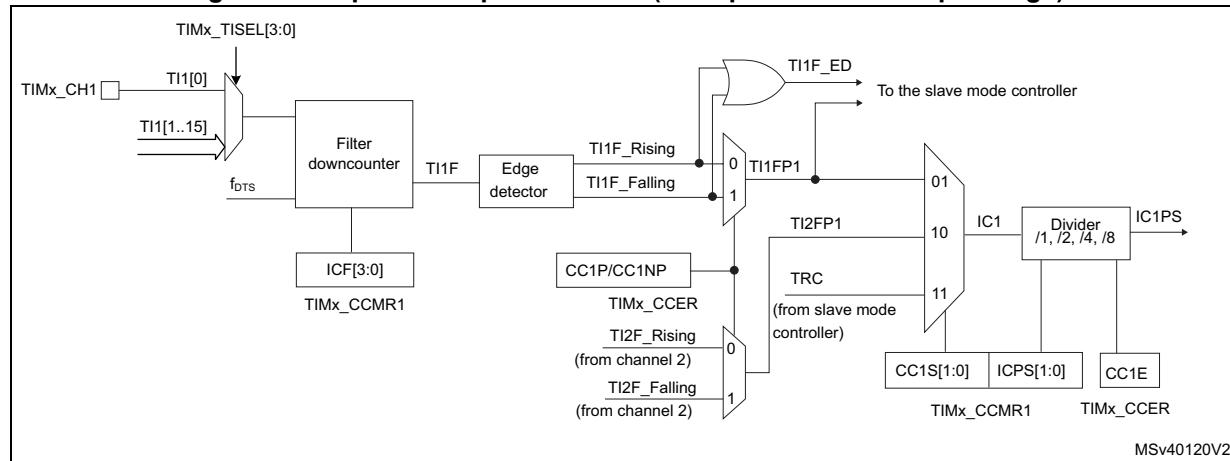
### 15.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

*Figure 83* to *Figure 86* give an overview of one Capture/Compare channel.

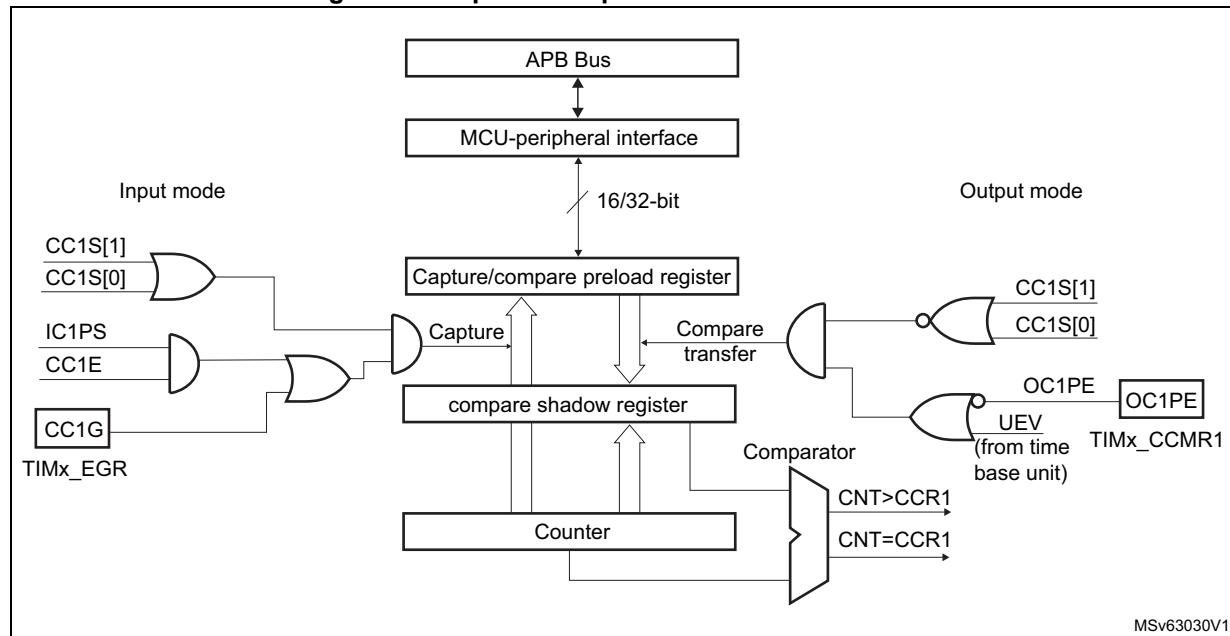
The input stage samples the corresponding TI<sub>x</sub> input to generate a filtered signal TI<sub>x</sub>F. Then, an edge detector with polarity selection generates a signal (TI<sub>x</sub>FP<sub>x</sub>) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (IC<sub>x</sub>PS).

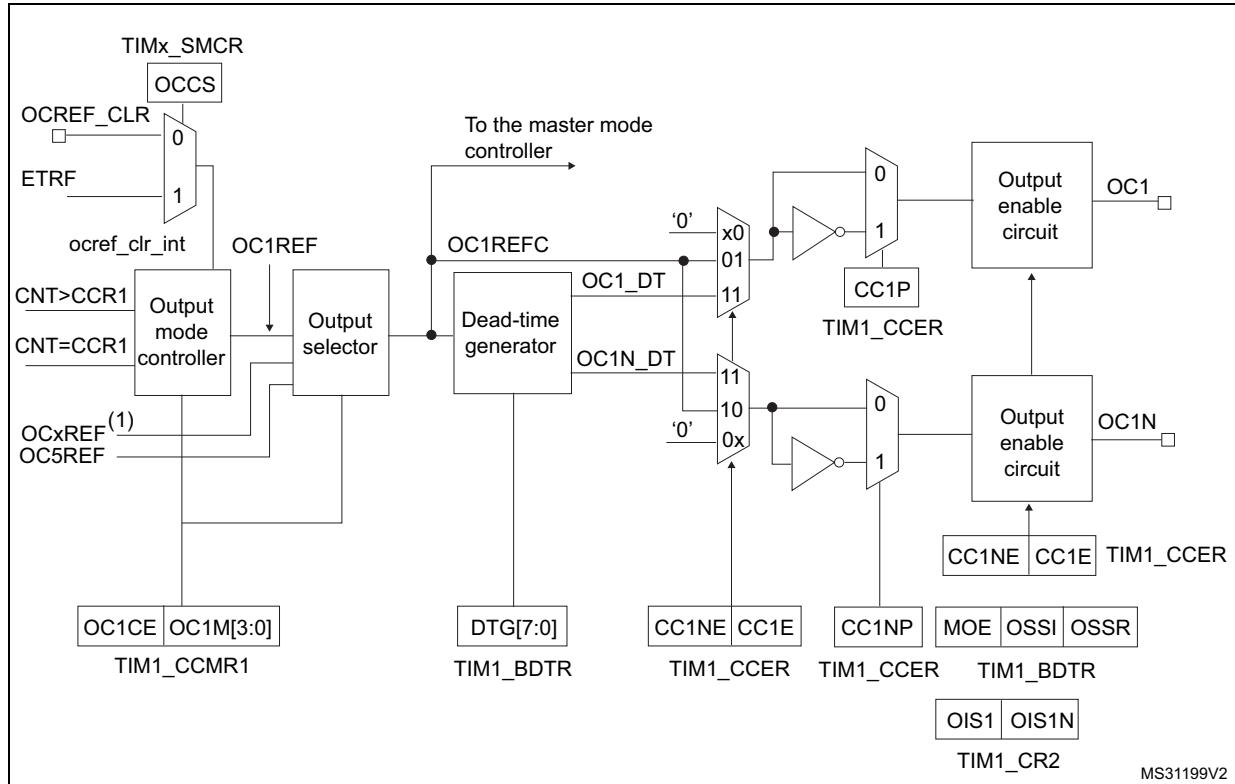
**Figure 83. Capture/compare channel (example: channel 1 input stage)**



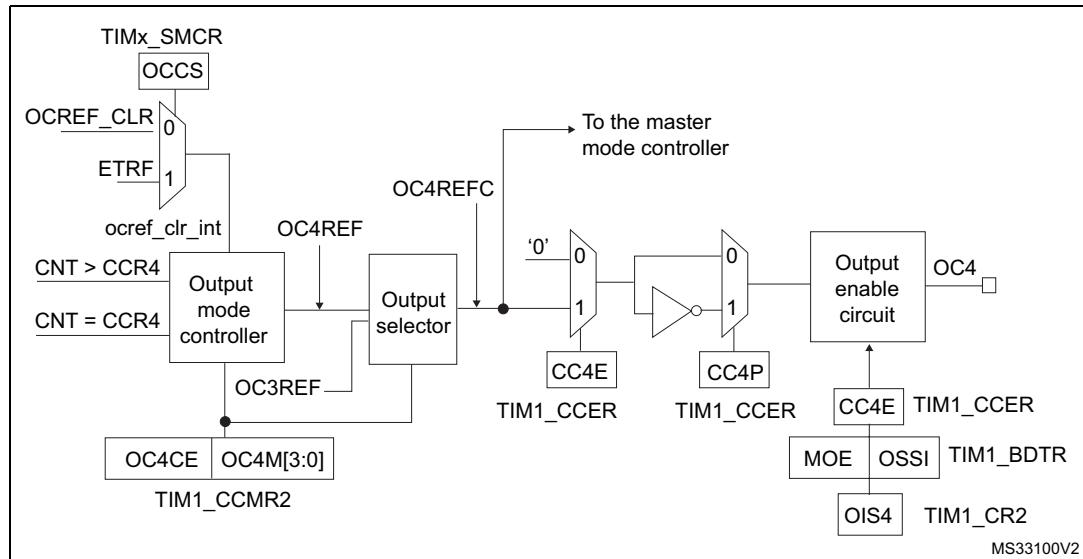
The output stage generates an intermediate waveform which is then used for reference: OC<sub>x</sub>Ref (active high). The polarity acts at the end of the chain.

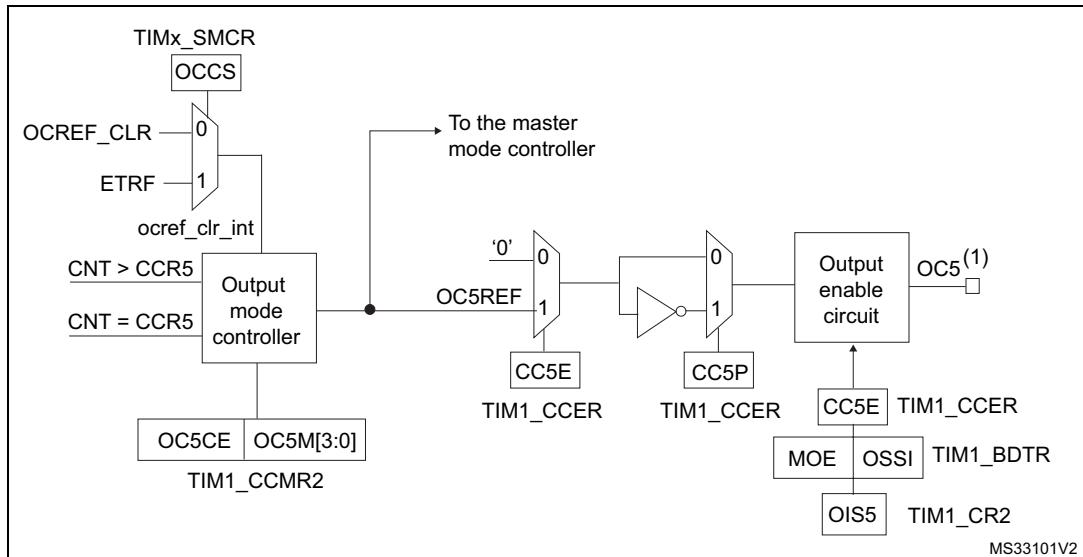
**Figure 84. Capture/compare channel 1 main circuit**



**Figure 85. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)**

1. OCxREF, where x is the rank of the complementary channel

**Figure 86. Output stage of capture/compare channel (channel 4)**

**Figure 87. Output stage of capture/compare channel (channel 5, idem ch. 6)**

1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 15.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when written with '0'.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.*

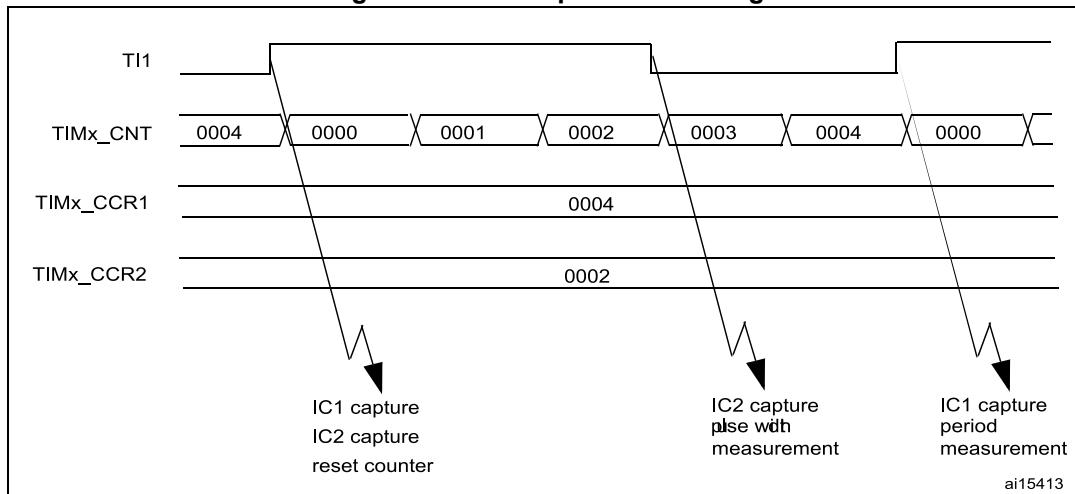
### 15.3.8 PWM input mode

This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
4. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP='10' (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx\_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx\_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

**Figure 88. PWM input mode timing**

### 15.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 15.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCXM=0001), be set inactive (OCXM=0010) or can toggle (OCXM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

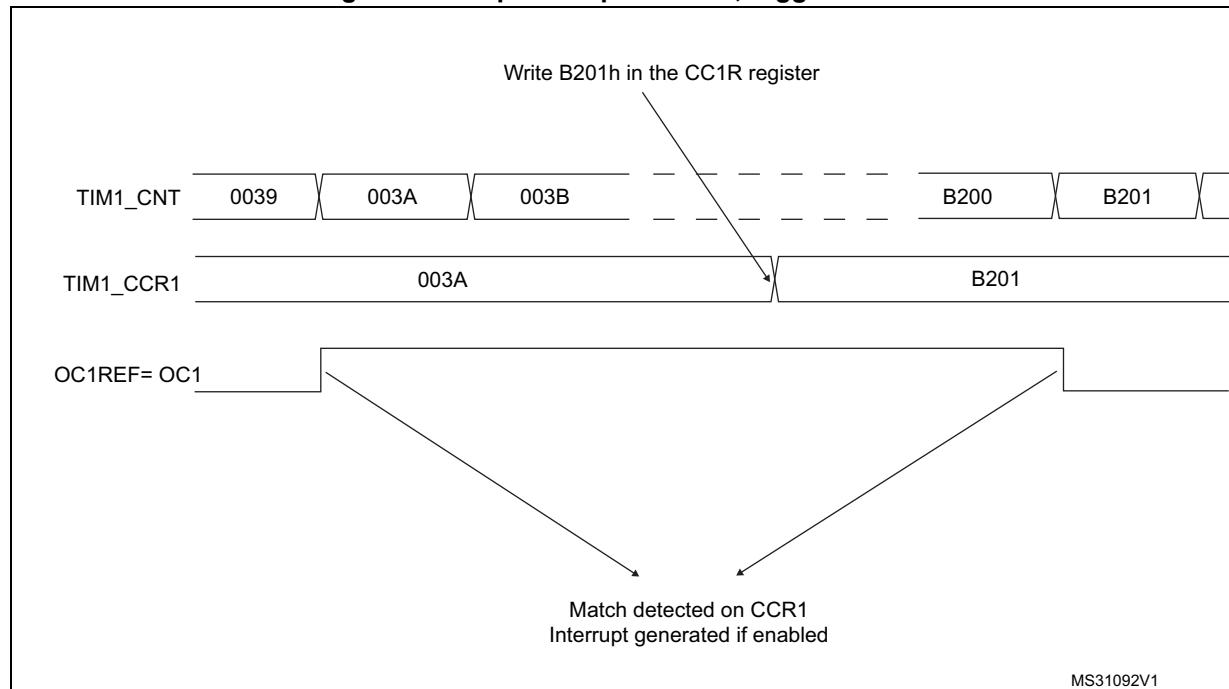
The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

#### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
  - Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write 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 89](#).

**Figure 89. Output compare mode, toggle on OC1**

MS31092V1

### 15.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 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

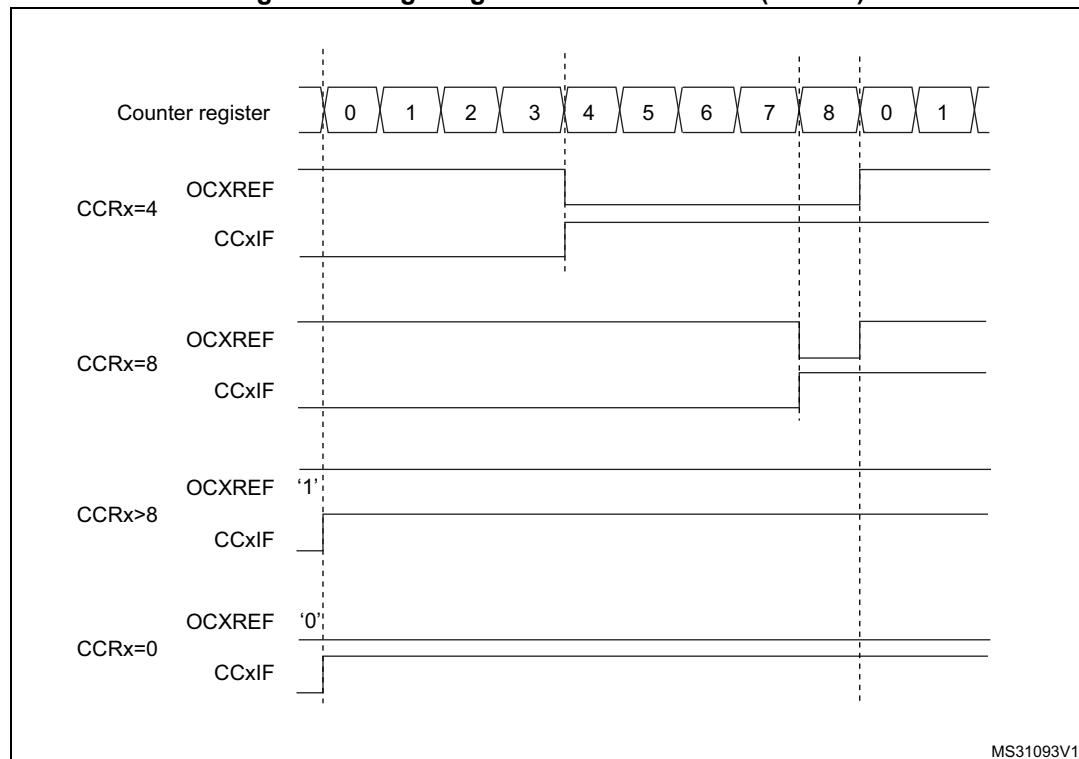
- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to the [Upcounting mode on page 302](#).

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as  $\text{TIMx\_CNT} < \text{TIMx\_CCR}_x$  else it becomes low. If the compare value in  $\text{TIMx\_CCR}_x$  is greater than the auto-reload value (in  $\text{TIMx\_ARR}$ ) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'.

[Figure 90](#) shows some edge-aligned PWM waveforms in an example where  $\text{TIMx\_ARR}=8$ .

**Figure 90. Edge-aligned PWM waveforms (ARR=8)**



- Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to the [Downcounting mode on page 306](#)

In PWM mode 1, the reference signal OCxRef is low as long as  $\text{TIMx\_CNT} > \text{TIMx\_CCR}_x$  else it becomes high. If the compare value in  $\text{TIMx\_CCR}_x$  is greater than the auto-reload value in  $\text{TIMx\_ARR}$ , then OCxREF is held at '1'. 0% PWM is not possible in this mode.

### PWM center-aligned mode

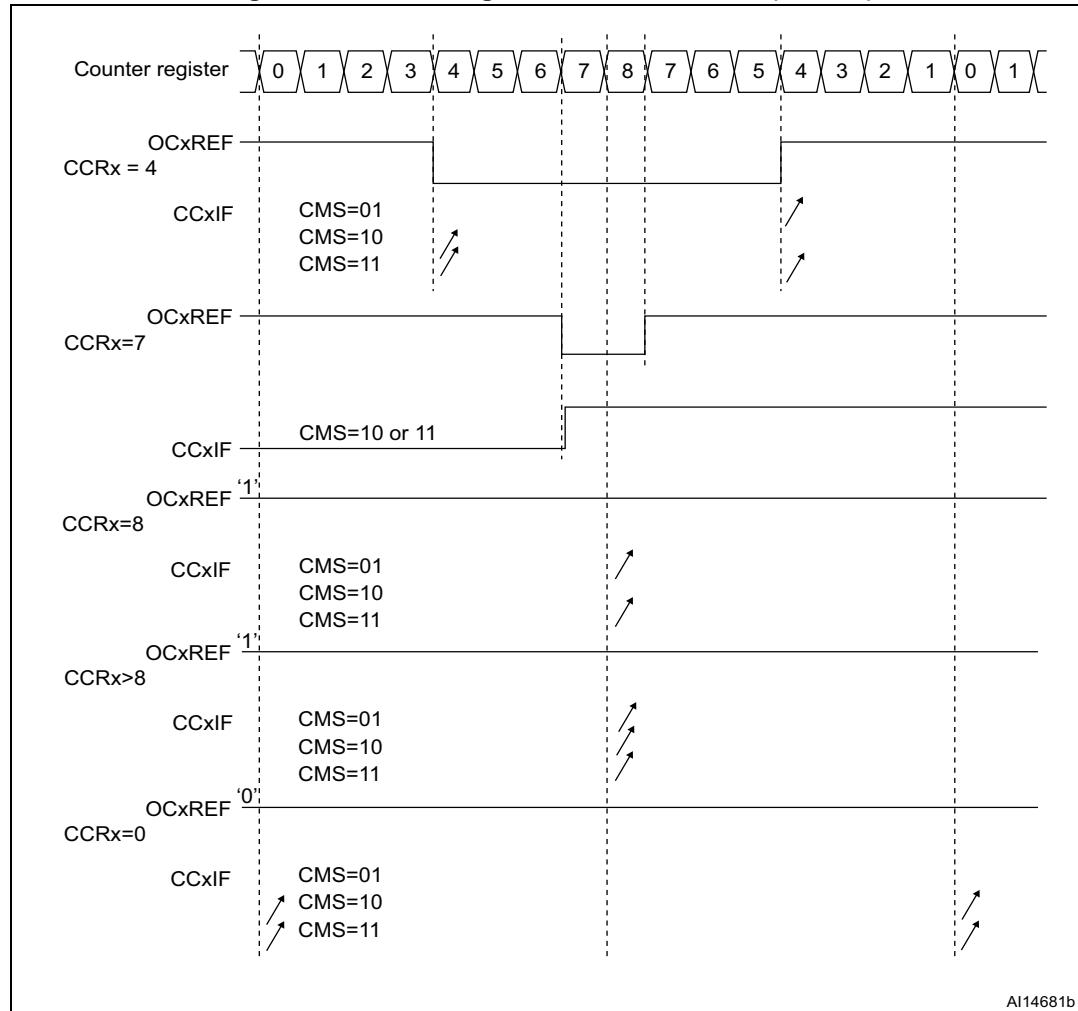
Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to the [Center-aligned mode \(up/down counting\) on page 309](#).

*Figure 91* 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 91. Center-aligned PWM waveforms (ARR=8)**



AI14681b

#### Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

- in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx\_CNT>TIMx\_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx\_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

### 15.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx\_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

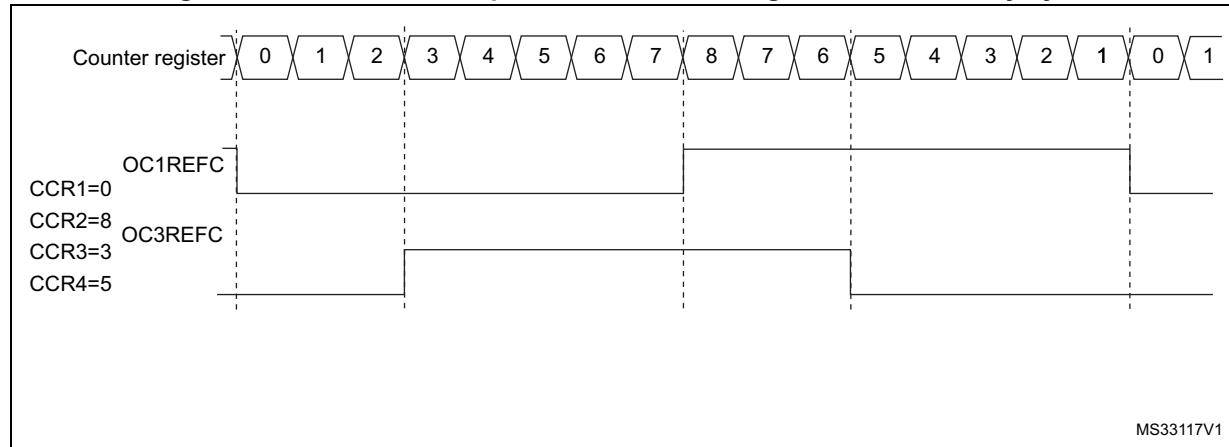
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

*Note:*

*The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

*Figure 92* represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

**Figure 92. Generation of 2 phase-shifted PWM signals with 50% duty cycle**

### 15.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and delay are determined by the two TIMx\_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

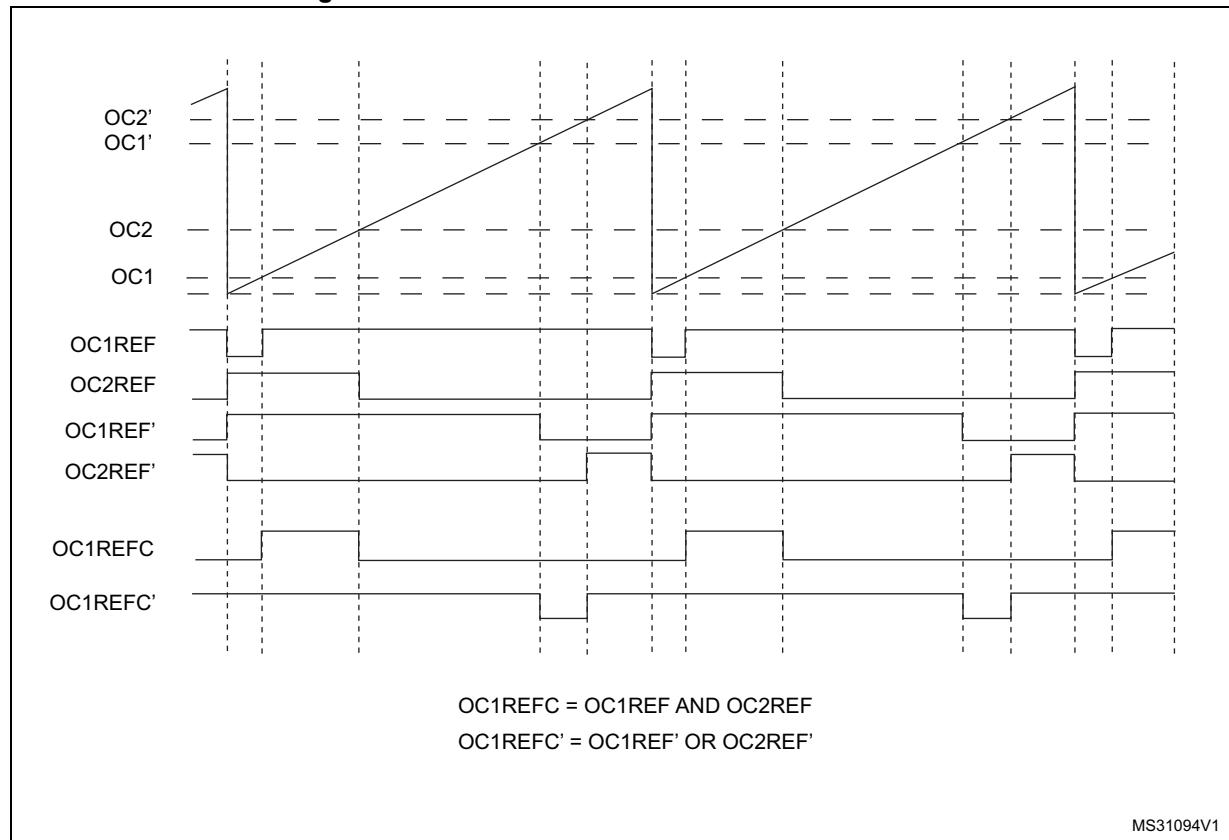
Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** *The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

**Figure 93** represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

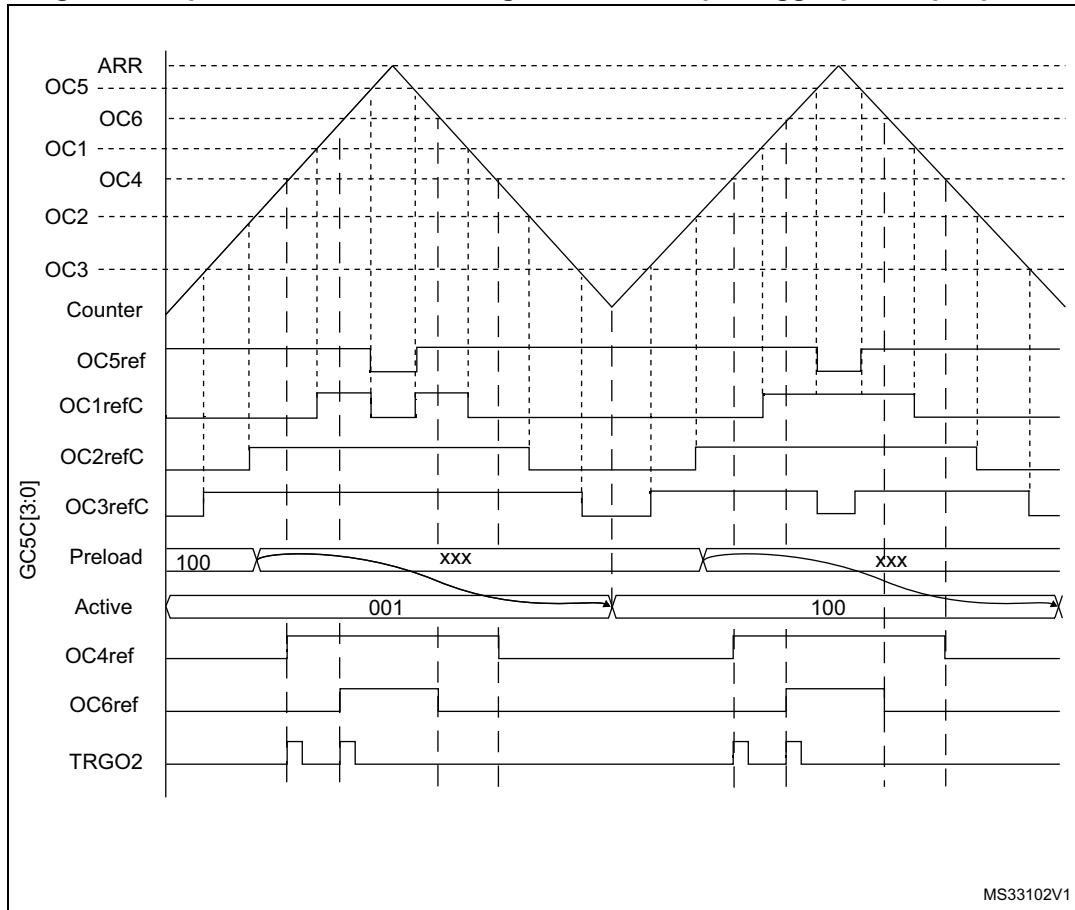
**Figure 93. Combined PWM mode on channel 1 and 3**

### 15.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx\_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, OC1REFC is controlled by TIMx\_CCR1 and TIMx\_CCR5
- If GC5C2 is set, OC2REFC is controlled by TIMx\_CCR2 and TIMx\_CCR5
- If GC5C3 is set, OC3REFC is controlled by TIMx\_CCR3 and TIMx\_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

**Figure 94. 3-phase combined PWM signals with multiple trigger pulses per period**

The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to [Section 15.3.27: ADC synchronization](#) for more details.

### 15.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OC<sub>x</sub> or complementary OC<sub>xN</sub>) can be selected independently for each output. This is done by writing to the CC<sub>xP</sub> and CC<sub>xNP</sub> bits in the TIM<sub>x</sub>\_CCER register.

The complementary signals OC<sub>x</sub> and OC<sub>xN</sub> are activated by a combination of several control bits: the CC<sub>xE</sub> and CC<sub>xNE</sub> bits in the TIM<sub>x</sub>\_CCER register and the MOE, OIS<sub>x</sub>, OIS<sub>xN</sub>, OSS<sub>I</sub> and OSS<sub>R</sub> bits in the TIM<sub>x</sub>\_BDTR and TIM<sub>x</sub>\_CR2 registers. Refer to

[Table 66: Output control bits for complementary OC<sub>x</sub> and OC<sub>xN</sub> channels with break feature on page 378](#) for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

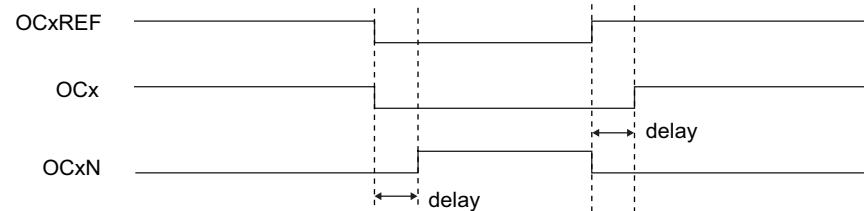
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

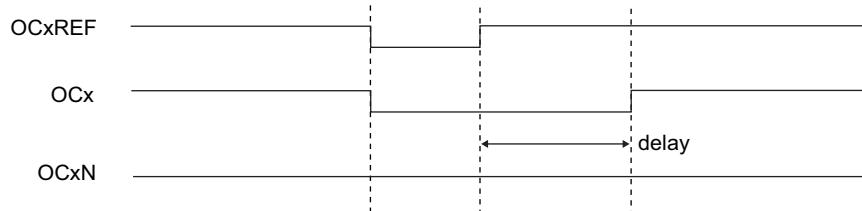
The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

**Figure 95. Complementary output with dead-time insertion**

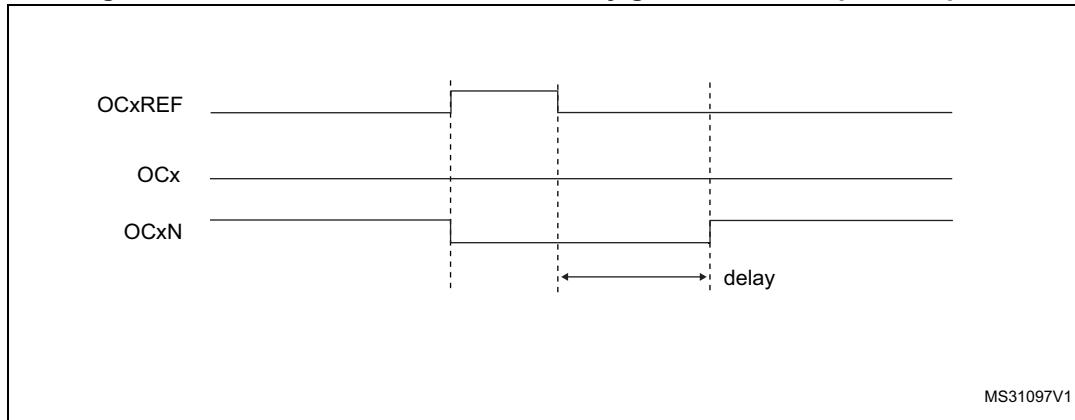


MS31095V1

**Figure 96. Dead-time waveforms with delay greater than the negative pulse**



MS31096V1

**Figure 97. Dead-time waveforms with delay greater than the positive pulse**

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx\_BDTR register. Refer to [Section 15.4.20: TIM1 break and dead-time register \(TIM1\\_BDTR\)](#) for delay calculation.

#### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx\_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

**Note:** When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

#### 15.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 timer. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx\_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSS1 bit in the TIMx\_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx\_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values.  
Refer to [Table 66: Output control bits for complementary OCx and OCxN channels with break feature on page 378](#) for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx\_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx\_OR2 and TIMx\_OR3 registers.

The sources for break (BRK) channel are:

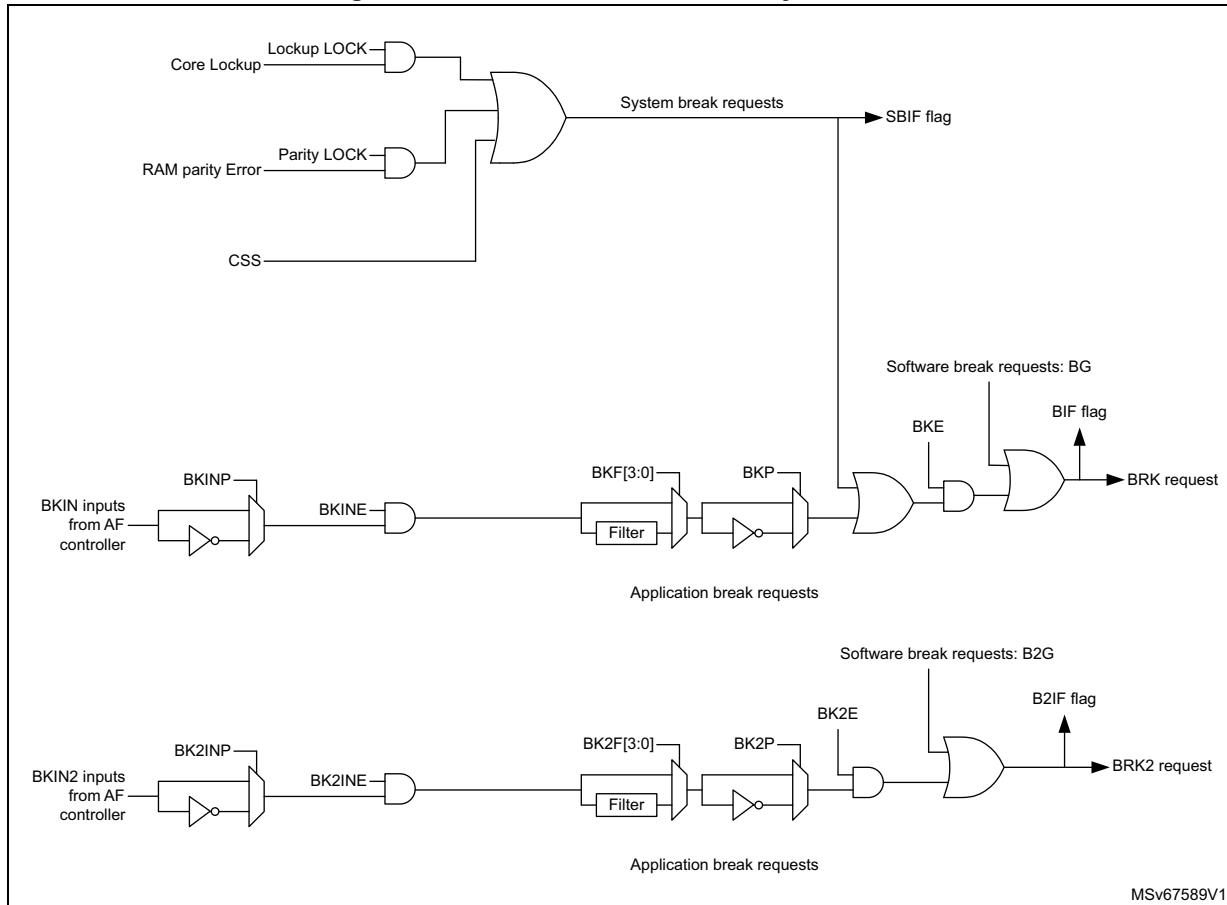
- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source:
  - the Cortex®-M0+ LOCKUP output
  - the SRAM parity error signal
  - a clock failure event generated by the CSS detector

The source for break2 (BRK2) is an external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering.

Break events can also be generated by software using BG and B2G bits in the TIMx\_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per [Figure 98](#) below.

Figure 98. Break and Break2 circuitry overview



**Note:** An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSS1 bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx\_CR2 register as soon as MOE=0. If OSS1=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their

active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck\_tim clock cycles).

- If OSS1=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (SBIF, BIF and B2IF bits in the TIMx\_SR register) is set. An interrupt is generated if the BIE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

**Note:** *If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSS1 value.*

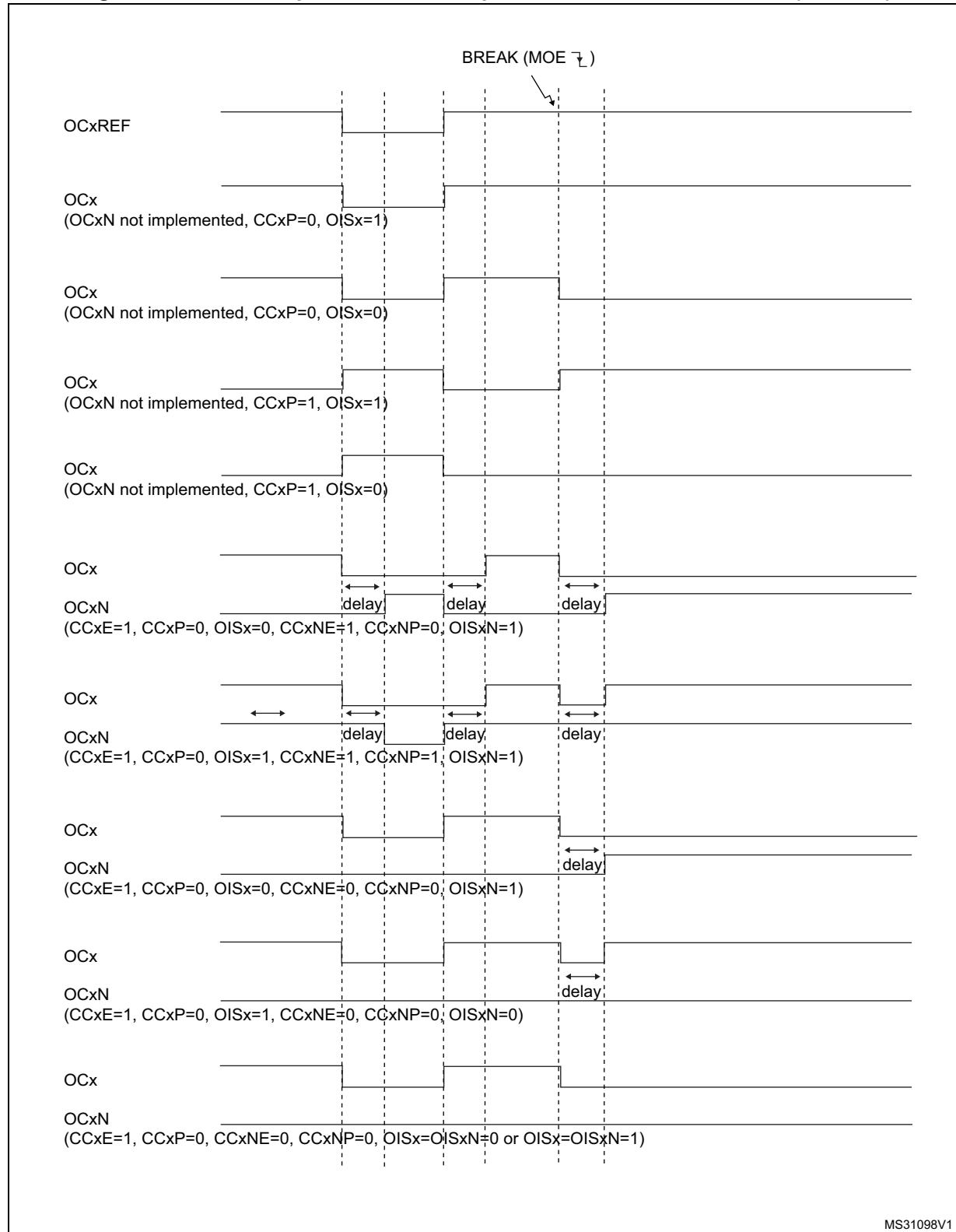
*If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx\_CR2 register.*

**Note:** *The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.*

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx\_BDTR register. Refer to [Section 15.4.20: TIM1 break and dead-time register \(TIM1\\_BDTR\)](#). The LOCK bits can be written only once after an MCU reset.

*Figure 99* shows an example of behavior of the outputs in response to a break.

Figure 99. Various output behavior in response to a break event on BRK (OSSI = 1)



MS31098V1

The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in [Table 62](#).

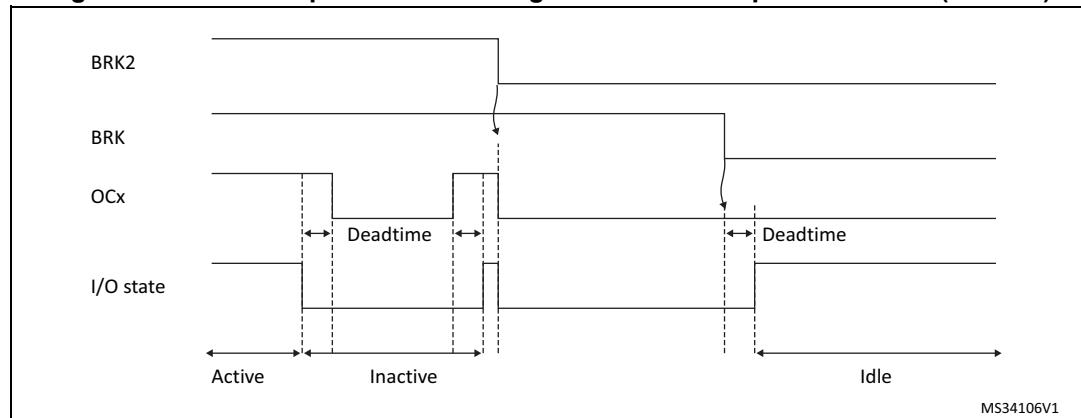
*Note:* BRK2 must only be used with OSSR = OSSI = 1.

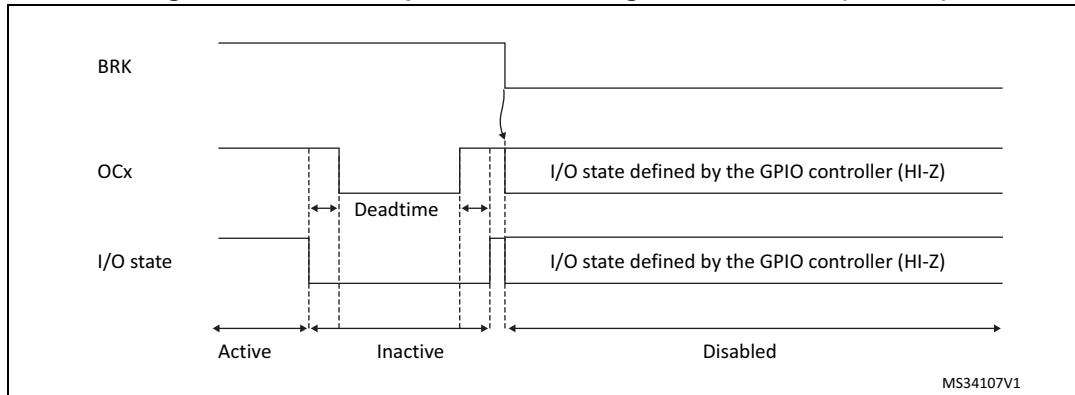
**Table 62. Behavior of timer outputs versus BRK/BRK2 inputs**

BRK	BRK2	Timer outputs state	Typical use case	
			OCxN output (low side switches)	OCx output (high side switches)
Active	X	<ul style="list-style-type: none"> <li>– Inactive then forced output state (after a deadtime)</li> <li>– Outputs disabled if OSSI = 0 (control taken over by GPIO logic)</li> </ul>	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

[Figure 100](#) gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx\_CCER register).

**Figure 100. PWM output state following BRK and BRK2 pins assertion (OSSI=1)**



**Figure 101. PWM output state following BRK assertion (OSSI=0)**

### 15.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on [Figure 102](#).

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBDID and BK2BDID bits in the TIMxBDTR register. The BKBDID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see [Table 63](#))

**Table 63. Break protection disarming conditions**

<b>MOE</b>	<b>BKDIR (BK2DIR)</b>	<b>BKDSRM (BK2DSRM)</b>	<b>Break protection state</b>
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

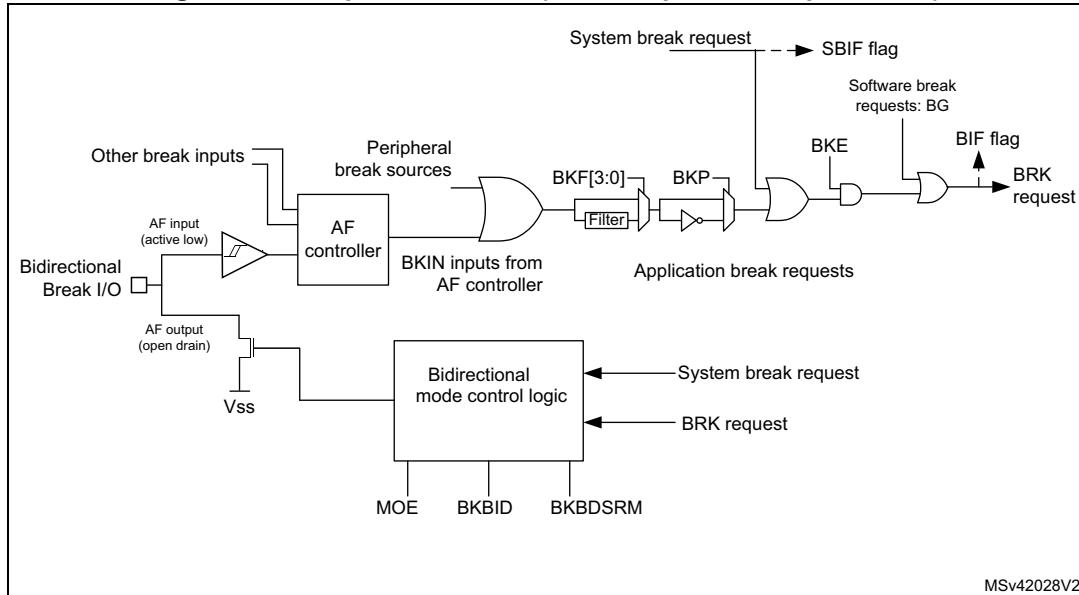
### Arming and re-arming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

**Figure 102. Output redirection (BRK2 request not represented)**

MSv42028V2

### 15.3.18 Clearing the OCxREF signal on an external event

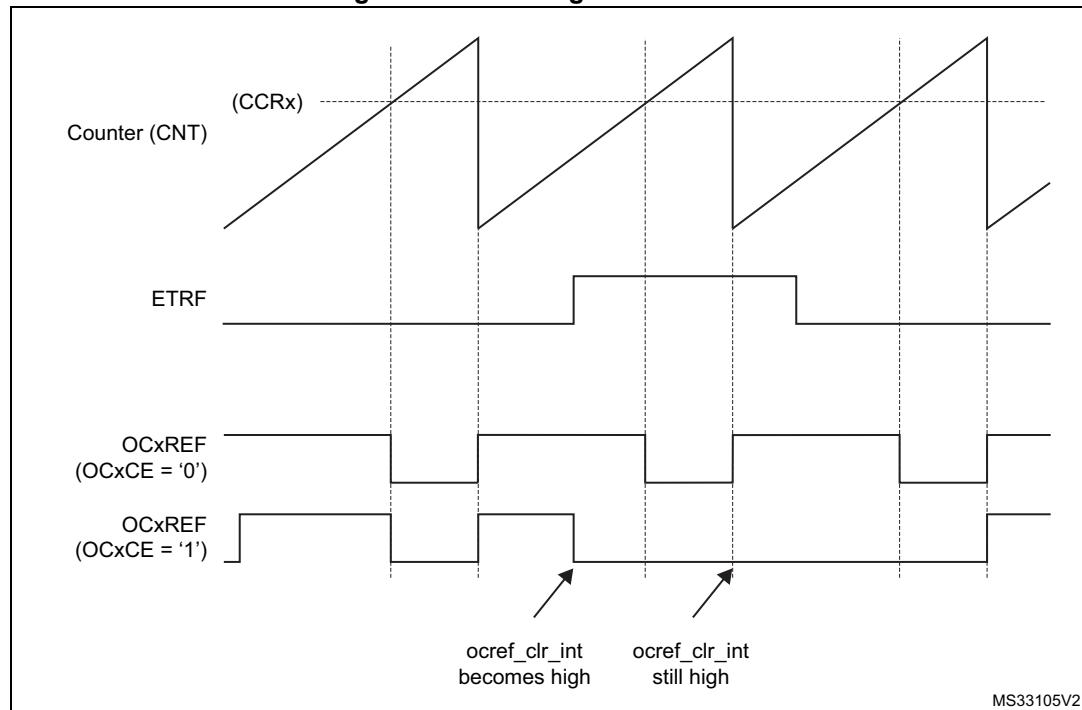
The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref\_clr\_int input (OCxCE enable bit in the corresponding TIMx\_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. ocref\_clr\_int input can be selected between the OCREF\_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx\_SMCR register.

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx\_SMCR register set to '00'.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx\_SMCR register set to '0'.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

*Figure 103* shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

**Figure 103. Clearing TIMx OCxREF**



Note:

*In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.*

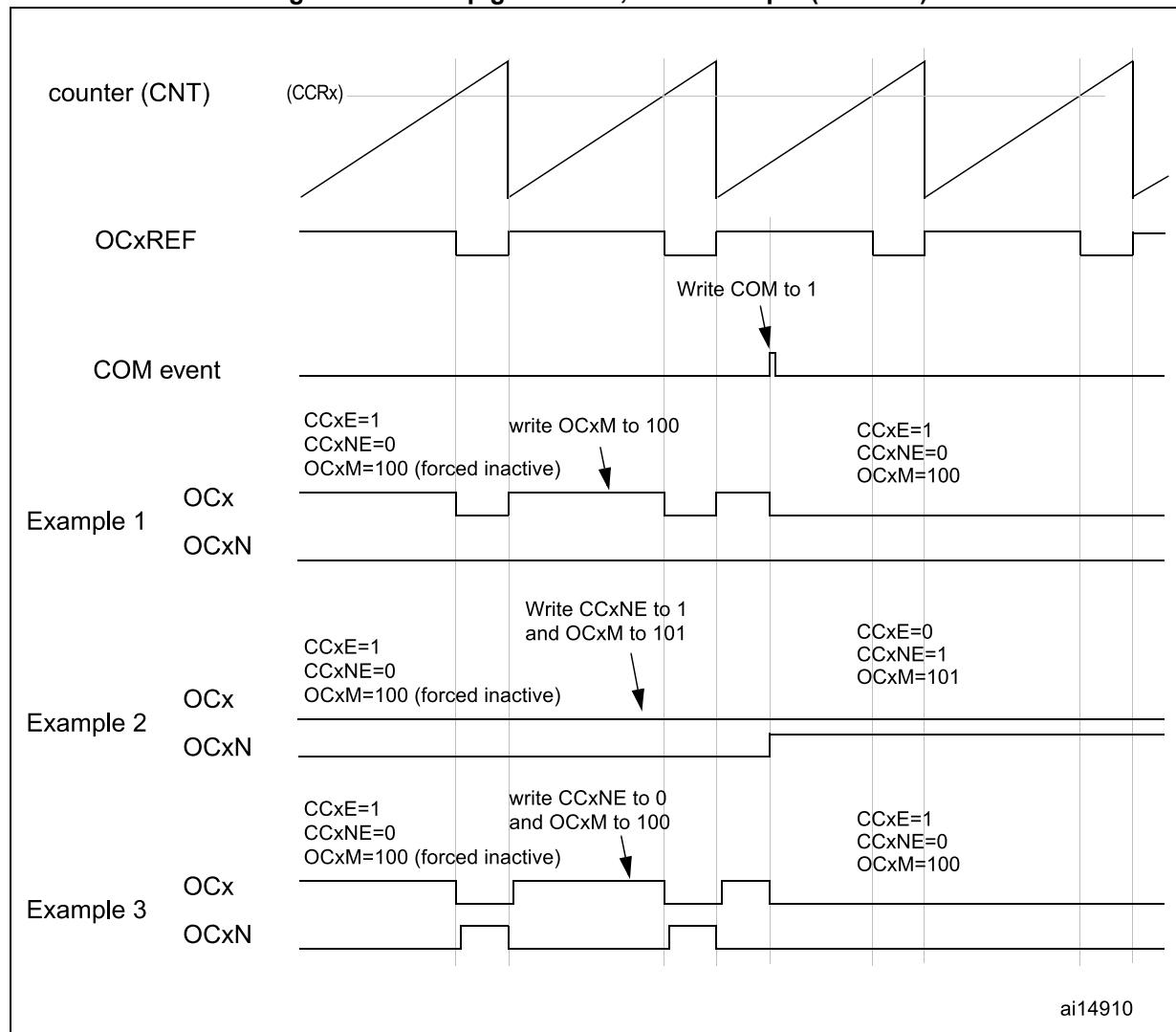
### 15.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx\_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx\_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx\_DIER register) or a DMA request (if the COMDE bit is set in the TIMx\_DIER register).

The [Figure 104](#) describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

**Figure 104. 6-step generation, COM example (OSSR=1)**



### 15.3.20 One-pulse mode

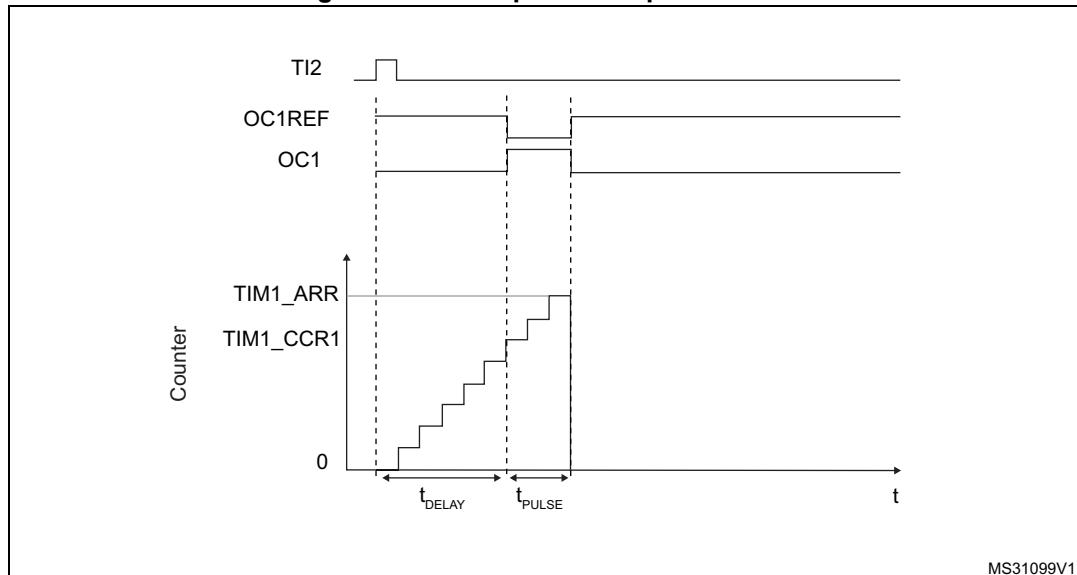
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx  $\leq$  ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

**Figure 105. Example of one pulse mode.**



MS31099V1

For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx\_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx\_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx\_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  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 one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx\_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case one has to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx\_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx\_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 15.3.21 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 15.3.20](#):

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

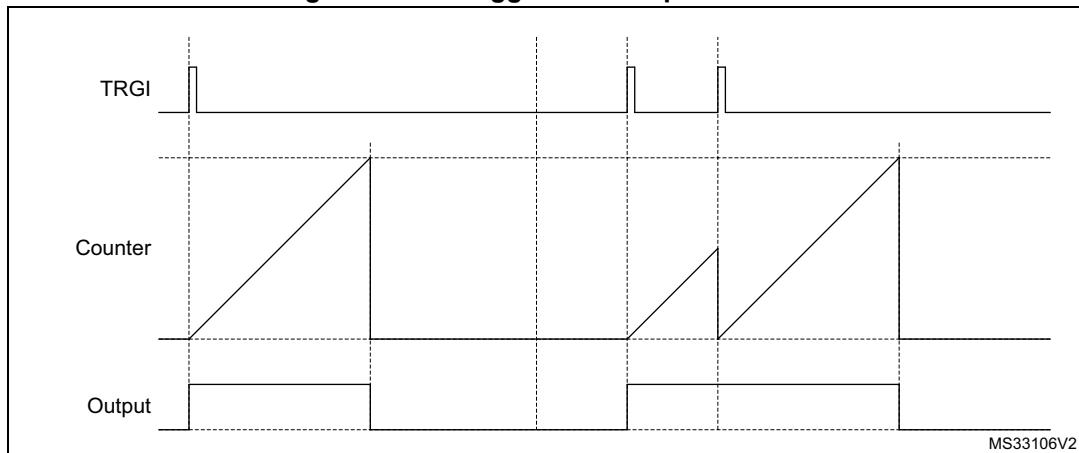
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx\_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx\_CR1.*

Figure 106. Retriggerable one pulse mode



### 15.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx\_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx\_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to [Table 64](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx\_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx\_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

*Note:*

*The prescaler must be set to zero when encoder mode is enabled*

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

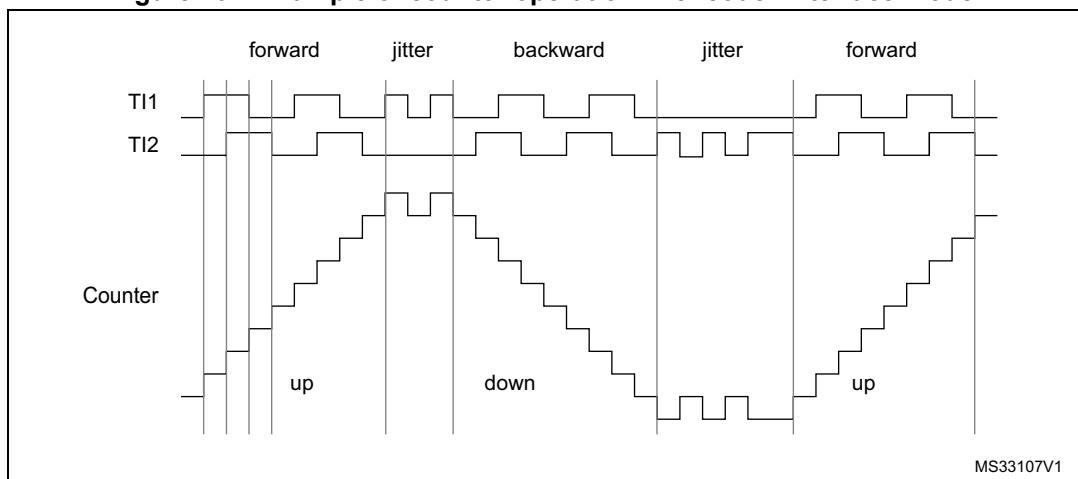
**Table 64. Counting direction versus encoder signals**

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

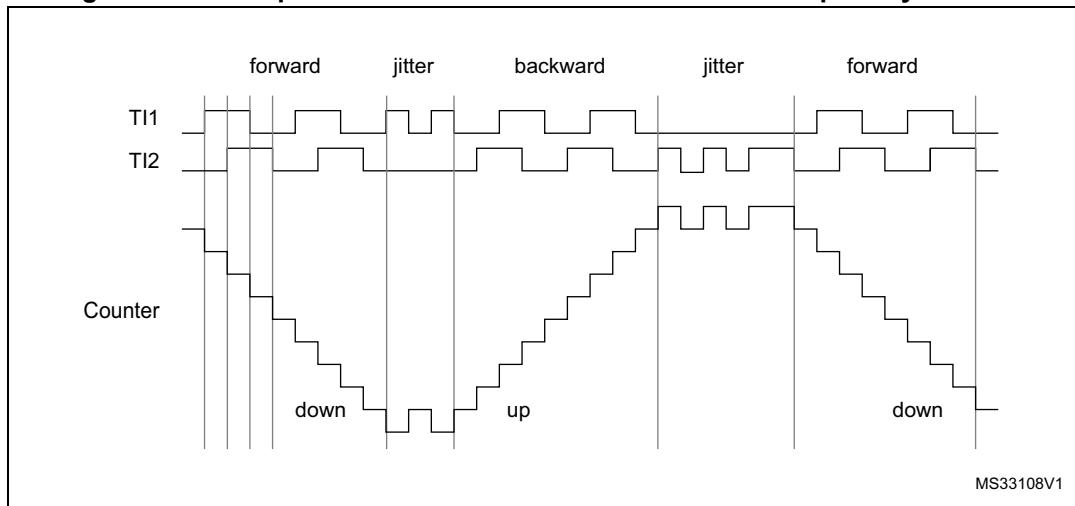
The [Figure 107](#) gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx\_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx\_CCMR2 register, TI1FP2 mapped on TI2).
- CC1P='0' and CC1NP='0' (TIMx\_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx\_CCER register, TI1FP2 non-inverted, TI1FP2=TI2).
- SMS='011' (TIMx\_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx\_CR1 register, Counter enabled).

**Figure 107. Example of counter operation in encoder interface mode.**

*Figure 108* gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

**Figure 108. Example of encoder interface mode with TI1FP1 polarity inverted.**



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

### 15.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

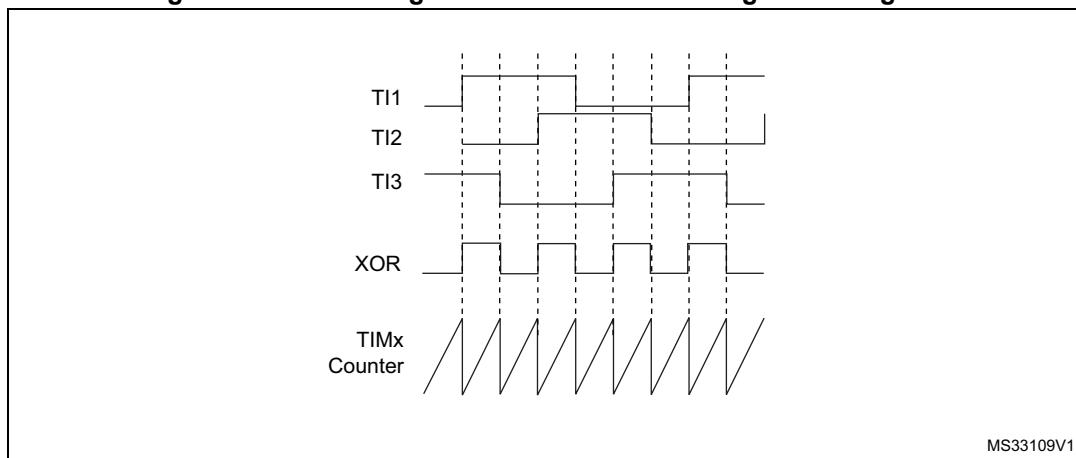
There is no latency between the UIF and UIFCPY flags assertion.

### 15.3.24 Timer input XOR function

The TI1S bit in the TIMx\_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx\_CH1, TIMx\_CH2 and TIMx\_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per [Figure 109](#) below.

**Figure 109. Measuring time interval between edges on 3 signals**



### 15.3.25 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM3) referred to as “interfacing timer” in [Figure 110](#). The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx\_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F\_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See [Figure 83: Capture/compare channel \(example: channel 1 input stage\) on page 320](#)). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1) through the TRGO output.

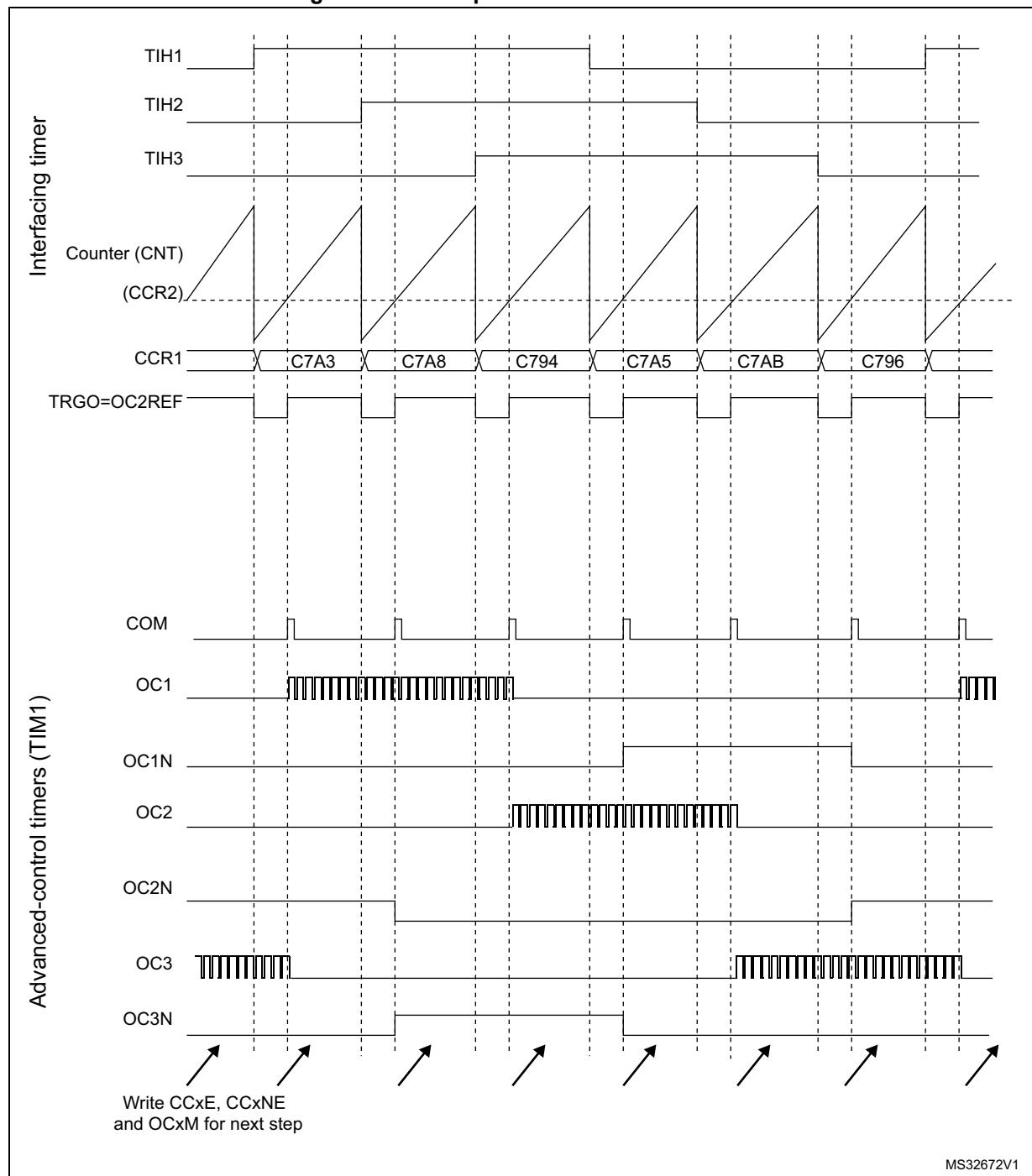
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx\_CR2 register to '1',
- Program the time base: write the TIMx\_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx\_CCMR1 register to '01'. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to '111' and the CC2S bits to '00' in the TIMx\_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx\_CR2 register to '101',

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx\_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx\_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The [Figure 110](#) describes this example.

Figure 110. Example of Hall sensor interface



### 15.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to [Section 16.3.19: Timer synchronization](#) for details. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

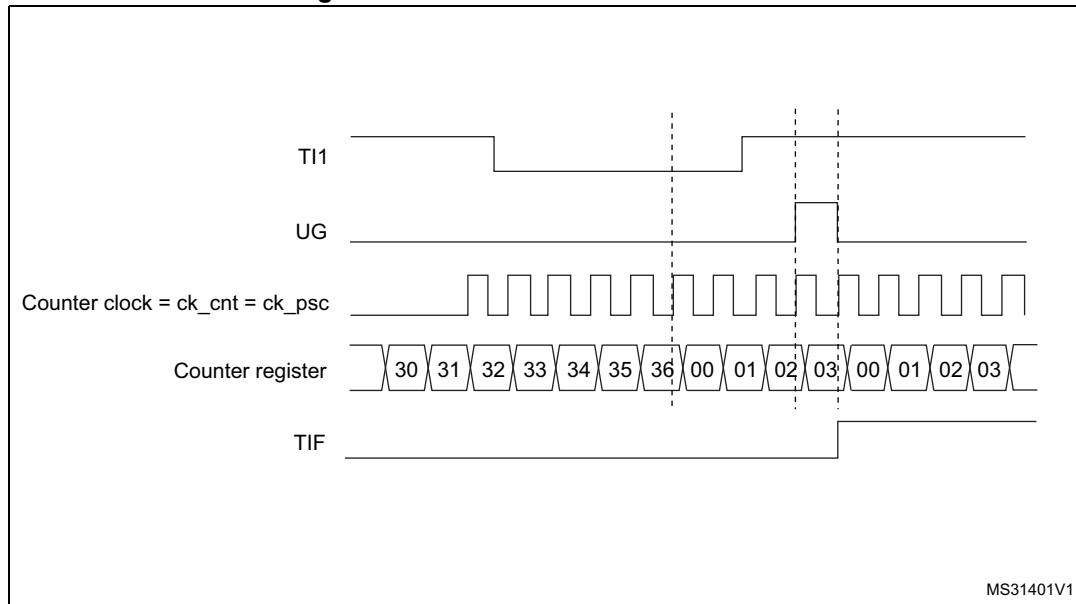
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx\_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
- Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

**Figure 111. Control circuit in reset mode**



### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

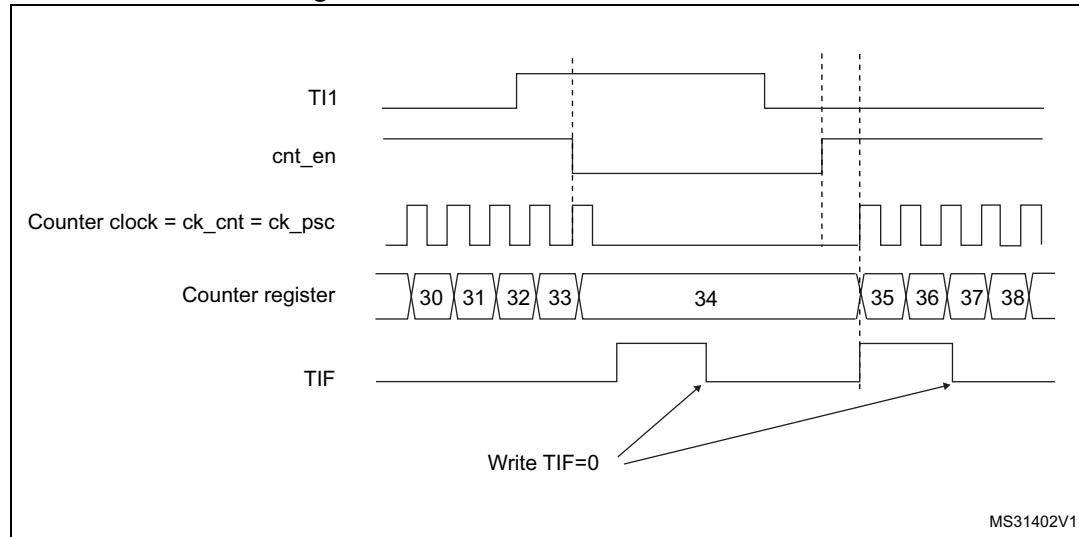
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx\_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 112. Control circuit in Gated mode**



### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx\_CCMR1

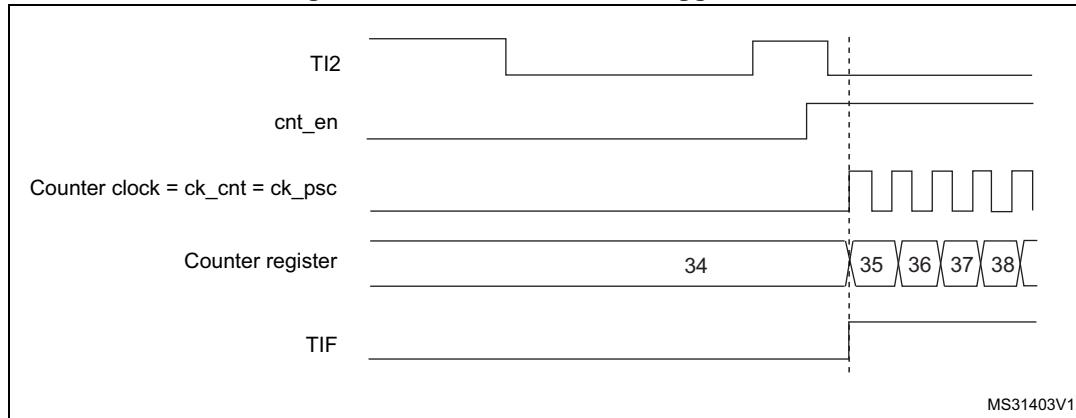
register. Write CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 113. Control circuit in trigger mode**



### Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

### Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register.

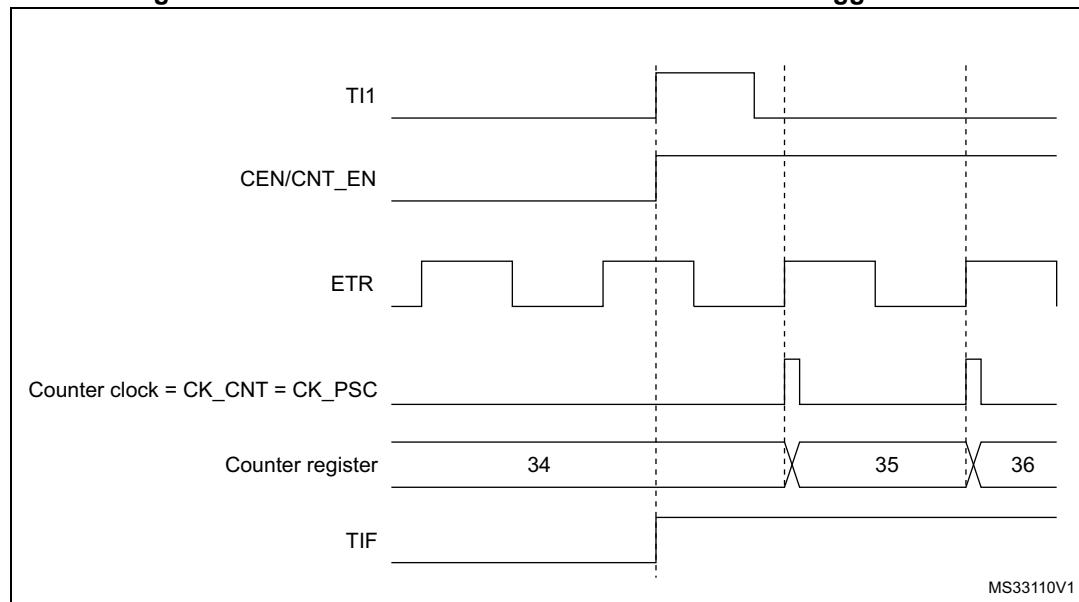
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS = 00: prescaler disabled
  - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S = 01 in TIMx\_CCMR1 register to select only the input capture source
  - CC1P = 0 and CC1NP = 0 in TIMx\_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

**Figure 114. Control circuit in external clock mode 2 + trigger mode**



Note:

*The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

### 15.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx\_CR2 register.

An example of an application for 3-phase motor drives is given in [Figure 94 on page 332](#).

**Note:** *The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

**Note:** *The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.*

### 15.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx\_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx\_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx\_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx\_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register:

Example:

00000: TIMx\_CR1

00001: TIMx\_CR2

00010: TIMx\_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers ( $x = 2, 3, 4$ ) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:  
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 15.3.29 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M0+ core halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBG module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

## 15.4 TIM1 registers

Refer to for a list of abbreviations used in register descriptions.

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

### 15.4.1 TIM1 control register 1 (TIM1\_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.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) used by the dead-time generators and the digital filters (ETR, TIx):

00:  $t_{DTS} = t_{CK\_INT}$

01:  $t_{DTS} = 2 * t_{CK\_INT}$

10:  $t_{DTS} = 4 * t_{CK\_INT}$

11: Reserved, do not program this value

*Note:*  $t_{DTS} = 1/f_{DTS}$ ,  $t_{CK\_INT} = 1/f_{CK\_INT}$ .

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx\_ARR register is not buffered

1: TIMx\_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set both when the counter is counting up or down.

*Note:* Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed

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.*

## 15.4.2 TIM1 control register 2 (TIM1\_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.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	
								rw	rw	rw	rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]				CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	

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

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

- 0000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.
- 0001: **Enable** - the Counter Enable signal CNT\_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx\_SMCR register).
- 0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.
- 0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).
- 0100: **Compare** - OC1REFC signal is used as trigger output (TRGO2)
- 0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)
- 0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)
- 0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)
- 1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)
- 1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)
- 1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2
- 1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2
- 1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2
- 1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2
- 1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2
- 1111: **Compare Pulse** - OC5REFC rising or OC6REFC falling edges generate pulses on TRGO2

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)

Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)

Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)

Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)

Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)

Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)

Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)

Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 7 **TI1S**: TI1 selection

0: The TIMx\_CH1 pin is connected to TI1 input

1: The TIMx\_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow selected information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT\_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC1REFC signal is used as trigger output (TRGO)

101: **Compare** - OC2REFC signal is used as trigger output (TRGO)

110: **Compare** - OC3REFC signal is used as trigger output (TRGO)

111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*

### 15.4.3 TIM1 slave mode control register (TIM1\_SMCR)

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.	TS[4:3]	Res.	Res.	Res.	SMS[3]	
										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or  $\overline{ETR}$  is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge.

1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).*

*It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).*

*If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of  $f_{CK\_INT}$  frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

- 00: Prescaler OFF
- 01: ETRP frequency divided by 2
- 10: ETRP frequency divided by 4
- 11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{DTS}$
- 0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2
- 0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4
- 0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8
- 0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6
- 0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8
- 0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6
- 0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8
- 1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6
- 1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8
- 1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5
- 1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6
- 1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8
- 1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5
- 1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6
- 1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

Bit 7 **MSM**: Master/slave mode

- 0: No action
- 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (ITR0)
- 00001: Internal Trigger 1 (ITR1)
- 00010: Internal Trigger 2 (ITR2)
- 00011: Internal Trigger 3 (ITR3)
- 00100: TI1 Edge Detector (TI1F\_ED)
- 00101: Filtered Timer Input 1 (TI1FP1)
- 00110: Filtered Timer Input 2 (TI2FP2)
- 00111: External Trigger input (ETRF)
- Others: Reserved

See [Table 65: TIM1 internal trigger connection on page 364](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source.

- 0: OCREF\_CLR\_INT is not connected (reserved configuration)
- 1: OCREF\_CLR\_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

*Note: The gated mode must not be used if TI1F\_ED is selected as the trigger input (TS=00100). Indeed, TI1F\_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.*

*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

**Table 65. TIM1 internal trigger connection**

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM1	Not connected	Not connected	TIM3	TIM17 OC1

#### 15.4.4 TIM1 DMA/interrupt enable register (TIM1\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

- 0: Trigger DMA request disabled
- 1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

- 0: COM DMA request disabled
- 1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable

- 0: CC4 DMA request disabled
- 1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable

- 0: CC3 DMA request disabled
- 1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable

- 0: CC2 DMA request disabled
- 1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable

- 0: CC1 DMA request disabled
- 1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable

- 0: Update DMA request disabled
- 1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable

- 0: Break interrupt disabled
- 1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable

- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable

- 0: COM interrupt disabled
- 1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable

- 0: CC4 interrupt disabled
- 1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable

- 0: CC3 interrupt disabled
- 1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

- 0: CC2 interrupt disabled
- 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 15.4.5 TIM1 status register (TIM1\_SR)

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.	CC6IF	CC5IF
														rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	SBIFF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0													

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag

Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag

Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIFF**: System Break interrupt flag

This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.

This flag must be reset to re-start PWM operation.

0: No break event occurred.

1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag

Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag

Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag

Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx\_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx\_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the content of TIMx\_CCR1 is greater than the content of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx\_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx\_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by a trigger event (refer to [Section 15.4.3: TIM1 slave mode control register \(TIM1\\_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

**15.4.6 TIM1 event generation register (TIM1\_EGR)**

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG						

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

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

*Note: This bit acts only on channels having a complementary output.*

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

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. The prescaler internal counter is also cleared (the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx\_ARR) if DIR=1 (downcounting).

### 15.4.7 TIM1 capture/compare mode register 1(TIM1\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Input capture mode:**

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
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
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 15:12 **IC2F[3:0]**: Input capture 2 filter

Refer to IC1F[3:0] description.

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Refer to IC1PSC[1:0] description.

**Bits 9:8 CC2S[1:0]: Capture/Compare 2 selection**

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

**Bits 7:4 IC1F[3:0]: Input capture 1 filter**

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{DTS}$

0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2

0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4

0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8

0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6

0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8

0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6

0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8

1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6

1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8

1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5

1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6

1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8

1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5

1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6

1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

**Bits 3:2 IC1PSC[1:0]: Input capture 1 prescaler**

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

**Bits 1:0 CC1S[1:0]: Capture/Compare 1 Selection**

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

### 15.4.8 TIM1 capture/compare mode register 1 [alternate] (TIM1\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

#### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output Compare 2 clear enable

Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode

Refer to OC1M[3:0] description.

Bit 11 **OC2PE**: Output Compare 2 preload enable

Refer to OC1PE description.

Bit 10 **OC2FE**: Output Compare 2 fast enable

Refer to OC1FE description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: OC1Ref is not affected by the ocref\_clr\_int signal

1: OC1Ref is cleared as soon as a High level is detected on ocref\_clr\_int signal  
(OCREF\_CLR input or ETRF input)

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0') as long as TIMx\_CNT>TIMx\_CCR1 else active (OC1REF='1').

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active. In downcounting, channel 1 is active as long as TIMx\_CNT>TIMx\_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note:* These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).

*Note:* In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

*Note:* On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

*Note:* The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

- 0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.
- 1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx\_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

- 0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
- 1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCCE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

### 15.4.9 TIM1 capture/compare mode register 2 (TIM1\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

**Input capture mode:**

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
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
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 15:12 **IC4F[3:0]**: Input capture 4 filter

Refer to IC1F[3:0] description.

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Refer to IC1PSC[1:0] description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx\_CCER).*

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Refer to IC1F[3:0] description.

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Refer to IC1PSC[1:0] description.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx\_CCER).*

### 15.4.10 TIM1 capture/compare mode register 2 [alternate] (TIM1\_CCMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

#### Output compare mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

- Bit 15 **OC4CE**: Output compare 4 clear enable  
Refer to OC1CE description.

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode

Refer to OC3M[3:0] description.

- Bit 11 **OC4PE**: Output compare 4 preload enable  
Refer to OC1PE description.

- Bit 10 **OC4FE**: Output compare 4 fast enable  
Refer to OC1FE description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx\_CCER).*

- Bit 7 **OC3CE**: Output compare 3 clear enable  
Refer to OC1CE description.

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

Refer to OC1M[3:0] description.

- Bit 3 **OC3PE**: Output compare 3 preload enable  
Refer to OC1PE description.

- Bit 2 **OC3FE**: Output compare 3 fast enable  
Refer to OC1FE description.

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx\_CCER).*

### 15.4.11 TIM1 capture/compare enable register (TIM1\_CCER)

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.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 21 **CC6P**: Capture/Compare 6 output polarity

Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable

Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity

Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable

Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity

Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity

Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable

Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity

Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable

Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity

Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable

Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity

Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable

Refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity  
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable  
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

**CC1 channel configured as output:**

- 0: OC1N active high.
- 1: OC1N active low.

**CC1 channel configured as input:**

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (channel configured as output).*

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

- 0: Off - OC1N is not active. OC1N level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 1 **CC1P**: Capture/Compare 1 output polarity

- 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
  - 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
- When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: The configuration is reserved, it must not be used.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 66](#) for details.

*Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

**Table 66. 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	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	X	0	X	X	Output disabled (not driven by the timer: Hi-Z).	
		0	0	0		
		0	1	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered).  Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). <b>Note:</b> BRK2 can only be used if OSSI = OSSR = 1.	
		1	0	0		
		1	1	1		
		1	1	1		

- When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

### 15.4.12 TIM1 counter (TIM1\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.														
r															
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

### 15.4.13 TIM1 prescaler (TIM1\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 15.4.14 TIM1 auto-reload register (TIM1\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 15.3.1: Time-base unit on page 300](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 15.4.15 TIM1 repetition counter register (TIM1\_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP\_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP\_CNT is reloaded with REP value only at the repetition update event U\_RC, any write to the TIMx\_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:  
the number of PWM periods in edge-aligned mode  
the number of half PWM period in center-aligned mode.

### 15.4.16 TIM1 capture/compare register 1 (TIM1\_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: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx\_CCR1 register is read-only and cannot be programmed.

### 15.4.17 TIM1 capture/compare register 2 (TIM1\_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\_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC2 output.

**If channel CC2 is configured as input:** CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx\_CCR2 register is read-only and cannot be programmed.

### 15.4.18 TIM1 capture/compare register 3 (TIM1\_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

**If channel CC3 is configured as output:** CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC3 output.

**If channel CC3 is configured as input:** CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx\_CCR3 register is read-only and cannot be programmed.

### 15.4.19 TIM1 capture/compare register 4 (TIM1\_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx\_CCR4 register is read-only and cannot be programmed.

### 15.4.20 TIM1 break and dead-time register (TIM1\_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2DSRM	BK2DSRM	BK2P	BK2E	BK2F[3:0]							
		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
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 BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSR, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

**Bit 28 BK<sub>BID</sub>:** Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BK<sub>BID</sub> bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

**Bit 27 BK<sub>2DSRM</sub>:** Break2 Disarm

Refer to BKDSRM description

**Bit 26 BK<sub>DSRM</sub>:** Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

**Bit 25 BK<sub>2P</sub>:** Break 2 polarity

- 0: Break input BRK2 is active low
- 1: Break input BRK2 is active high

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

**Bit 24 BK<sub>2E</sub>:** Break 2 enable

- 0: Break input BRK2 disabled
- 1: Break input BRK2 enabled

*Note: The BRK2 must only be used with OSSR = OSSI = 1.*

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK2 acts asynchronously
- 0001:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , 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

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, BRK acts asynchronously
- 0001:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , 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

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled

In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSS1 bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register).

See OC/OCN enable description for more details ([Section 15.4.11: TIM1 capture/compare enable register \(TIM1\\_CCER\)](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk\_acth and BKIN sources, as per [Figure 98: Break and Break2 circuitry overview](#)).

0: Break function disabled

1: Break function enabled

*Note:* This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 15.4.11: TIM1 capture/compare enable register \(TIM1\\_CCER\)](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note:* This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).

**Bit 10 OSS1:** Off-state selection for Idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details ([Section 15.4.11: TIM1 capture/compare enable register \(TIM1\\_CCER\)](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bits 9:8 LOCK[1:0]:** Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx\_BDTR register, OISx and OISxN bits in TIMx\_CR2 register and BK2BID, BKBD, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSS1, OSSR and DTG[7:0] bits in TIMx\_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx\_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSS1 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> = 2 x t<sub>DTS</sub>.

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t<sub>DTG</sub> with t<sub>DTG</sub> = 8 x t<sub>DTS</sub>.

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t<sub>DTG</sub> with t<sub>DTG</sub> = 16 x t<sub>DTS</sub>.

Example if t<sub>DTS</sub> = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 15.4.21 TIM1 DMA control register (TIM1\_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	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

**Example:** Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx\_CR1.

- If DBL = 7 bytes and DBA = TIMx\_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx\_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx\_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx\_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

- If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.
- If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

**Example:**

00000: TIMx\_CR1,  
00001: TIMx\_CR2,  
00010: TIMx\_SMCR,

...

### 15.4.22 TIM1 DMA address for full transfer (TIM1\_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx\_CR1 address) + (DBA + DMA index) x 4

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 15.4.23 TIM1 capture/compare mode register 3 (TIM1\_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

#### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Refer to OC1CE description.

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode

Refer to OC1M description.

Bit 11 **OC6PE**: Output compare 6 preload enable

Refer to OC1PE description.

Bit 10 **OC6FE**: Output compare 6 fast enable

Refer to OC1FE description.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE**: Output compare 5 clear enable

Refer to OC1CE description.

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode

Refer to OC1M description.

Bit 3 **OC5PE**: Output compare 5 preload enable

Refer to OC1PE description.

Bit 2 **OC5FE**: Output compare 5 fast enable

Refer to OC1FE description.

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

### 15.4.24 TIM1 capture/compare register 5 (TIM1\_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.												
rw	rw	rw													
15      14      13      12      11      10      9      8      7      6      5      4      3      2      1      0															
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3

Distortion on Channel 3 output:

0: No effect of OC5REF on OC3REFC

1: OC3REFC is the logical AND of OC3REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 30 **GC5C2**: Group Channel 5 and Channel 2

Distortion on Channel 2 output:

0: No effect of OC5REF on OC2REFC

1: OC2REFC is the logical AND of OC2REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bit 29 **GC5C1**: Group Channel 5 and Channel 1

Distortion on Channel 1 output:

0: No effect of OC5REF on OC1REFC5

1: OC1REFC is the logical AND of OC1REFC and OC5REF

This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

*Note: it is also possible to apply this distortion on combined PWM signals.*

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC5 output.

### 15.4.25 TIM1 capture/compare register 6 (TIM1\_CCR6)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC6 output.

### 15.4.26 TIM1 alternate function option register 1 (TIM1\_AF1)

Address offset: 0x60

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	Res	Res	Res	Res	Res	Res	ETRSEL[3:2]
															rw rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]	Res.	Res.	Res.	Res.	Res.	BKINP	Res.	BKINE							
rw	rw					rw									rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.

0000: ETR legacy mode

0011: ADC1 AWD1

0100: ADC1 AWD2

0101: ADC1 AWD3

Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

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

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note:* Refer to [Figure 77: TIM1 ETR input circuitry](#) and to [Figure 98: Break and Break2 circuitry overview](#).

#### 15.4.27 TIM1 Alternate function register 2 (TIM1\_AF2)

Address offset: 0x64

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.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BK2 INP	Res.	BK2INE							
						rw									rw

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

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note:* Refer to [Figure 98: Break and Break2 circuitry overview](#).

### 15.4.28 TIM1 timer input selection register (TIM1\_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[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
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input

0000: TIM1\_CH4 input

Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input

0000: TIM1\_CH3 input

Others: Reserved

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

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM1\_CH2 input

Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM1\_CH1 input

Others: Reserved

### 15.4.29 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 67. TIM1 register map and reset values**

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
0x00	<b>TIM1_CR1</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.	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	0	0	0			
0x04	<b>TIM1_CR2</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.	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	0	0	0	0		
0x08	<b>TIM1_SMCR</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.	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	0	0	0	0	0	
0x0C	<b>TIM1_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.	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	0	0	0	0	0	0
0x10	<b>TIM1_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.	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	<b>TIM1_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.	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	0	0	0	0	0	0
0x18	<b>TIM1_CCMR1</b> Output Compare mode	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.	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	0	0	0	0	0	0
	<b>TIM1_CCMR1</b> Input Capture mode	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.	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	0	0	0	0	0	0
0x1C	<b>TIM1_CCMR2</b> Output Compare mode	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.	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	0	0	0	0	0	0
	<b>TIM1_CCMR2</b> Input Capture mode	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.	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	0	0	0	0	0	0
0x20	<b>TIM1_CCER</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.	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	0	0	0	0	0	0

**Table 67. TIM1 register map and reset values (continued)**

**Table 67. TIM1 register map and reset values (continued)**

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 16 General-purpose timer (TIM3)

### 16.1 TIM3 introduction

The general-purpose timer TIM3 consists of a 16-bit auto-reload counter driven by a programmable prescaler.

The timer may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

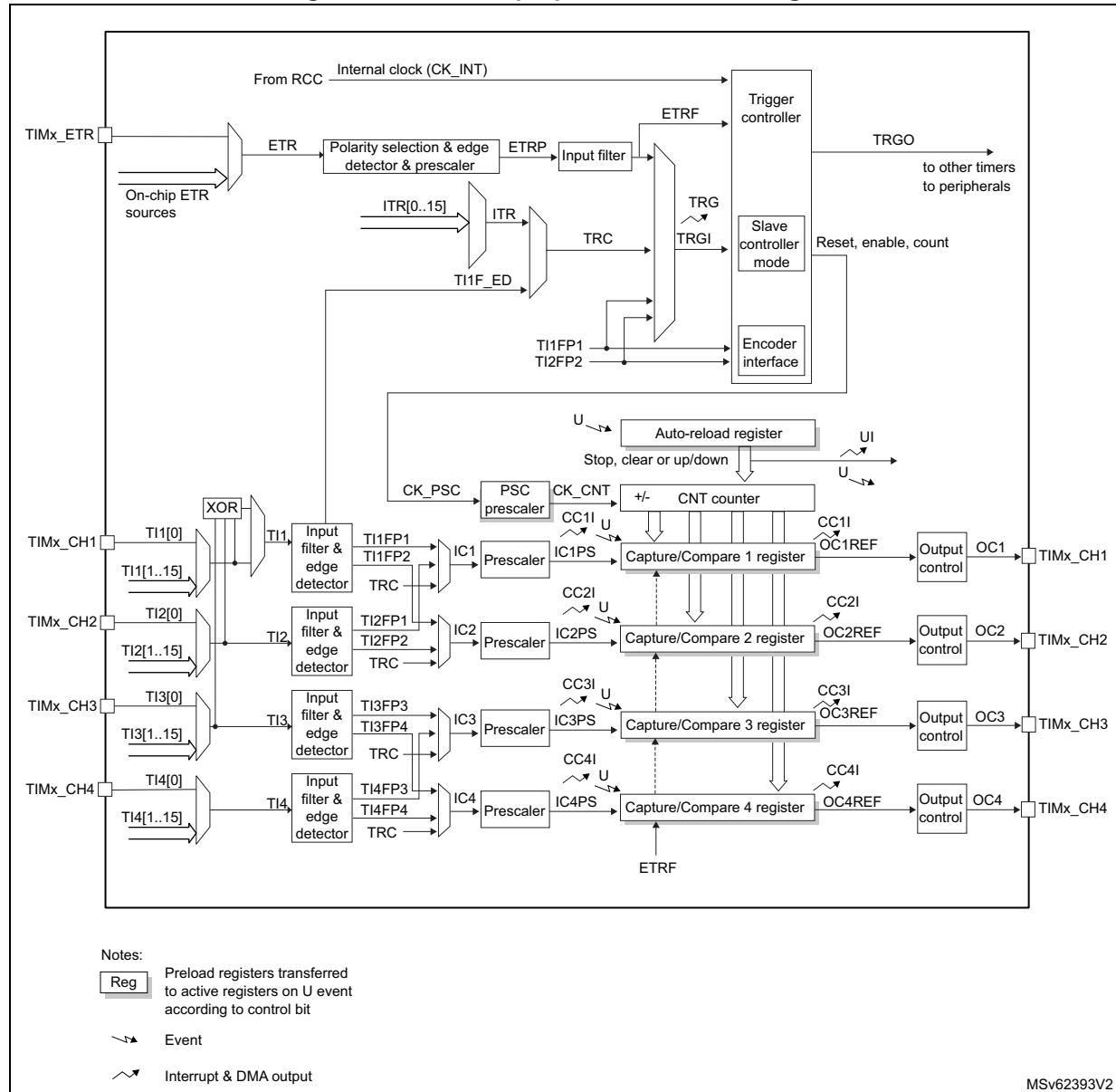
Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

### 16.2 TIM3 main features

General-purpose TIMx timer features include:

- 16-bit TIM3 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 115. General-purpose timer block diagram



## 16.3 TIM3 functional description

### 16.3.1 Time-base unit

The main block of the programmable timer is a 16-bit counter with its related auto-reload register. The counter can count up, 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.

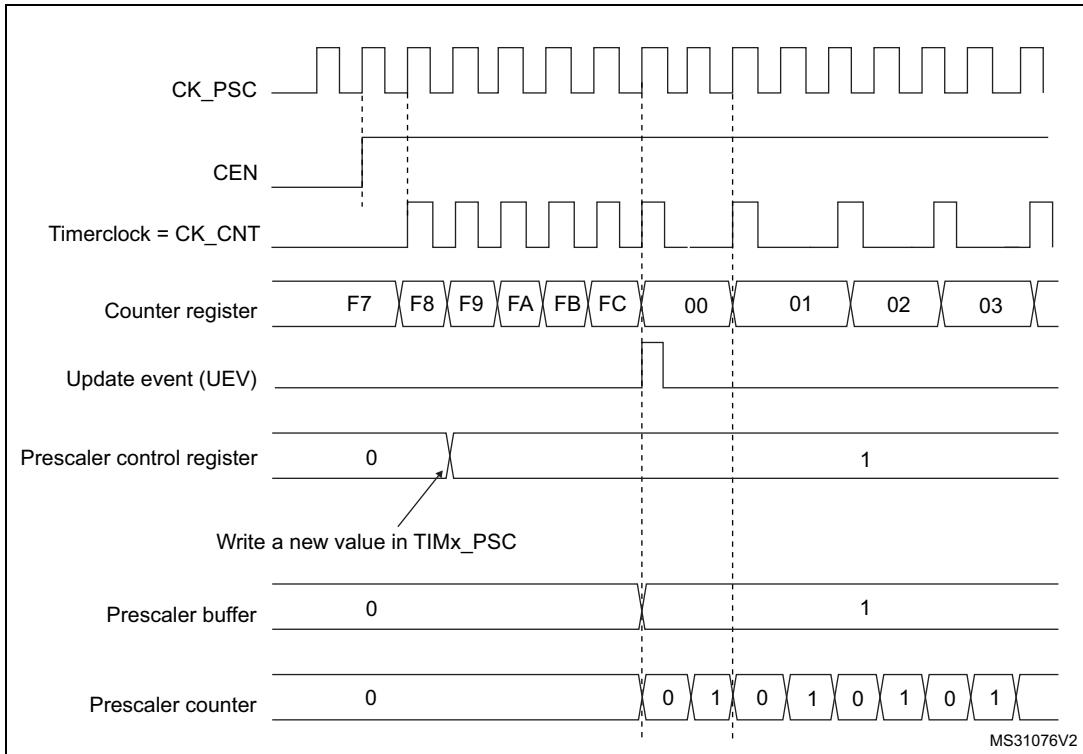
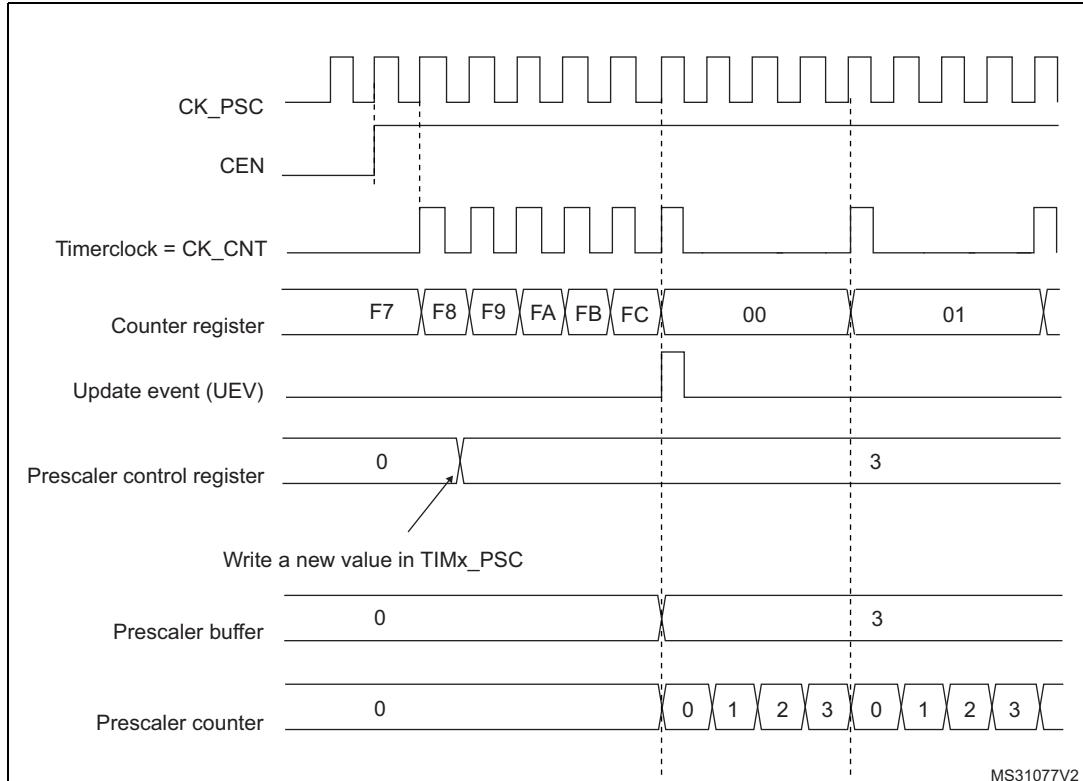
The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

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 116* and *Figure 117* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 116. Counter timing diagram with prescaler division change from 1 to 2****Figure 117. 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).

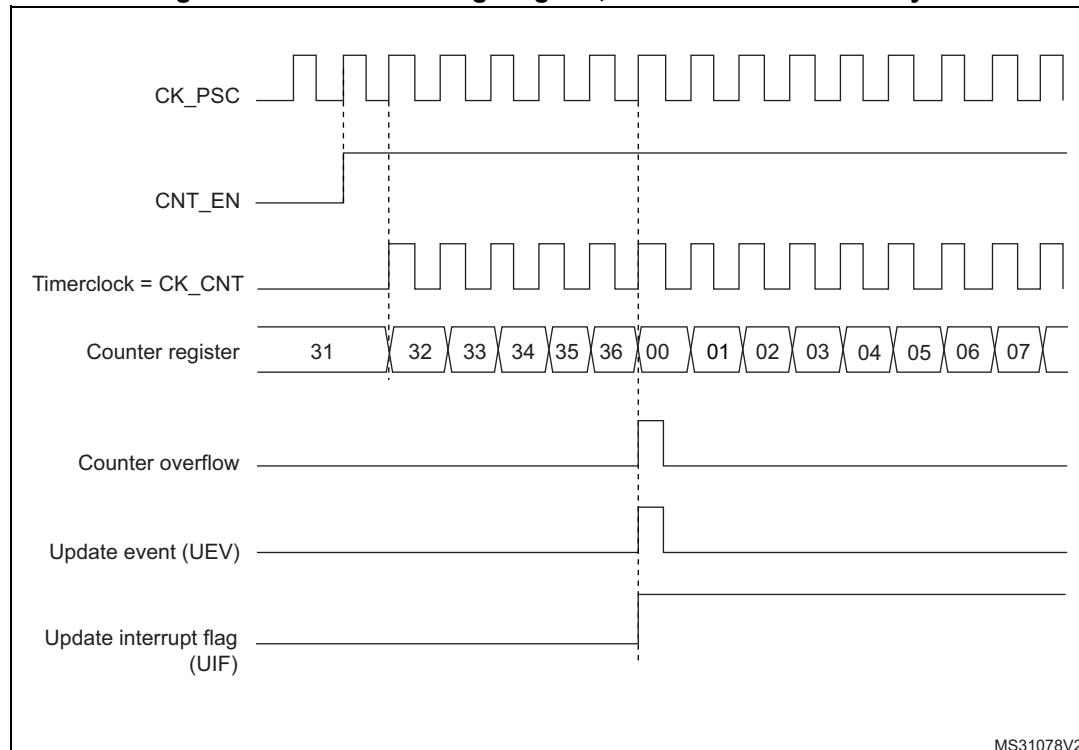
The UEV event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well 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.

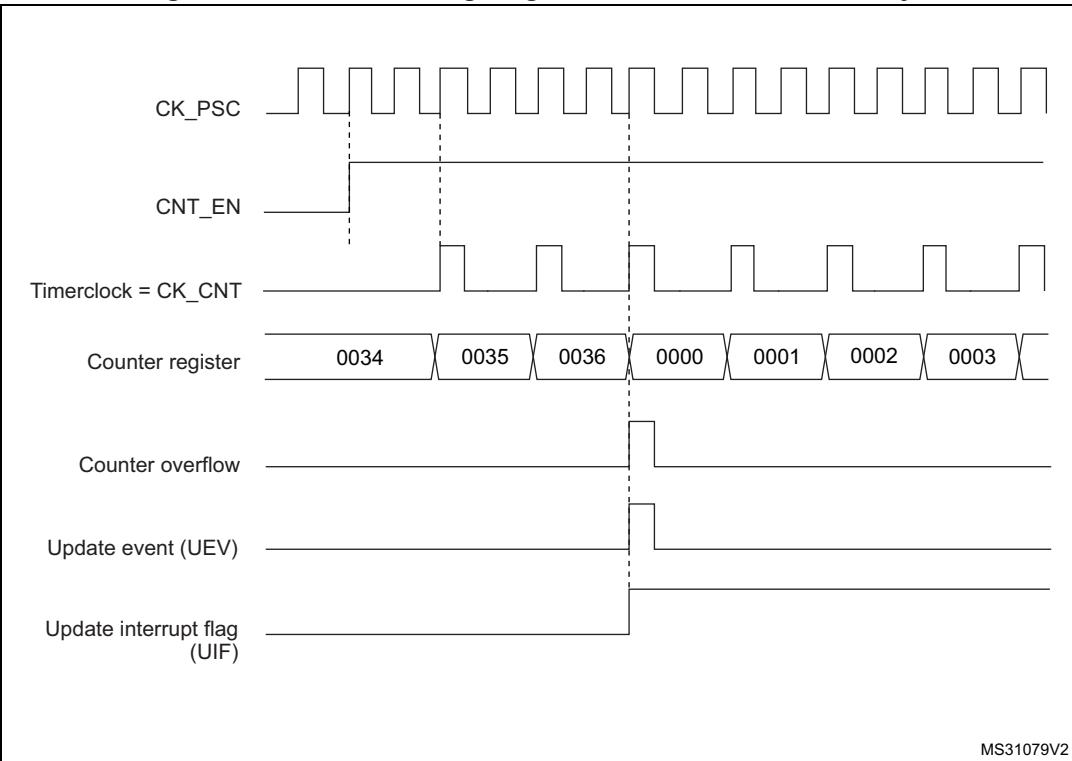
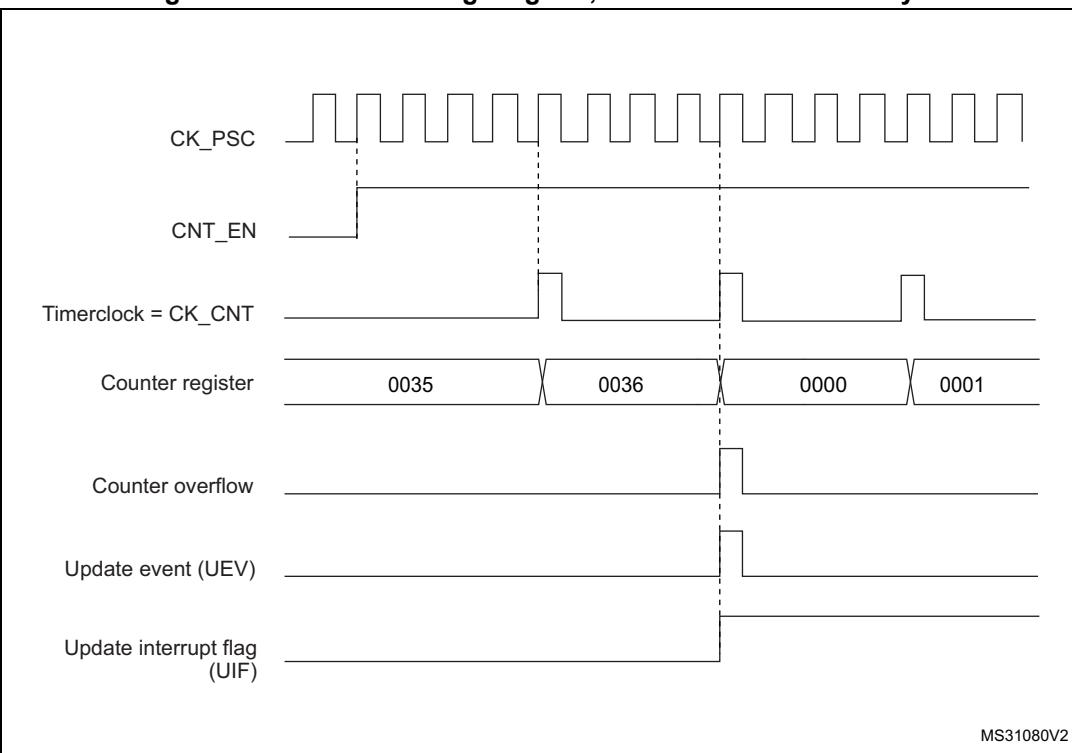
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

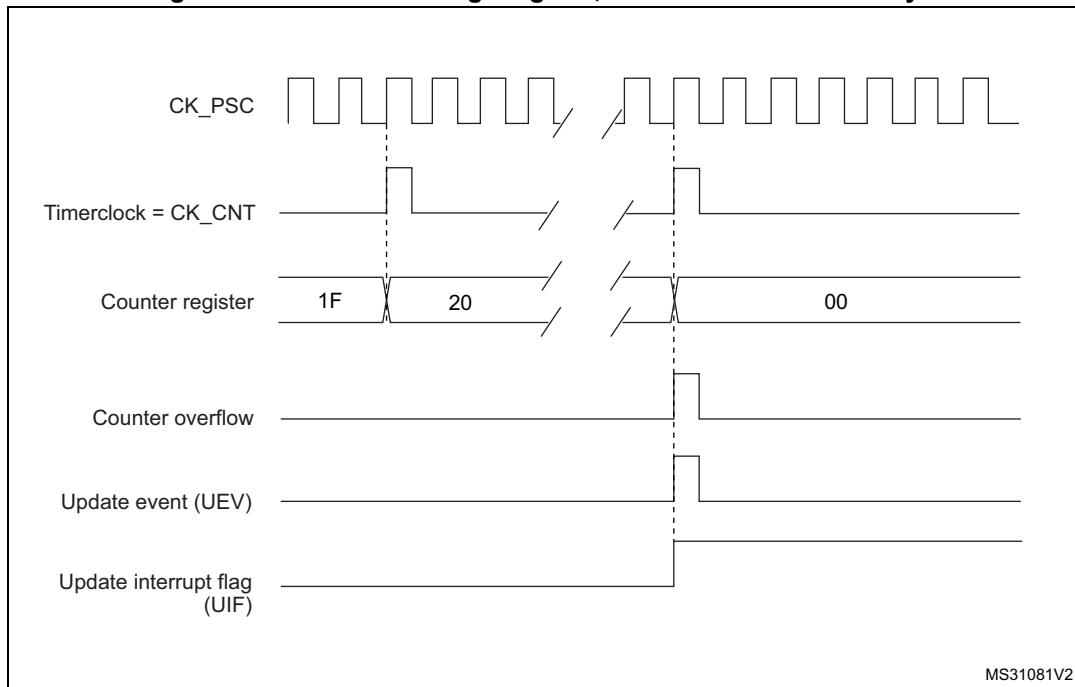
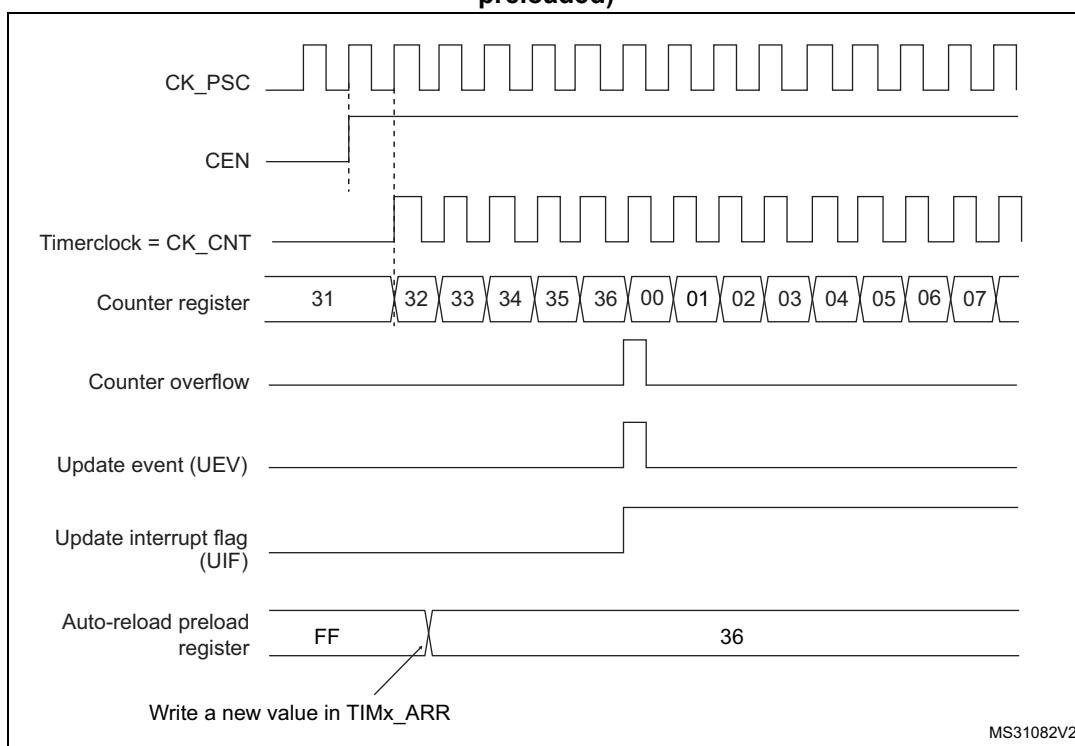
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

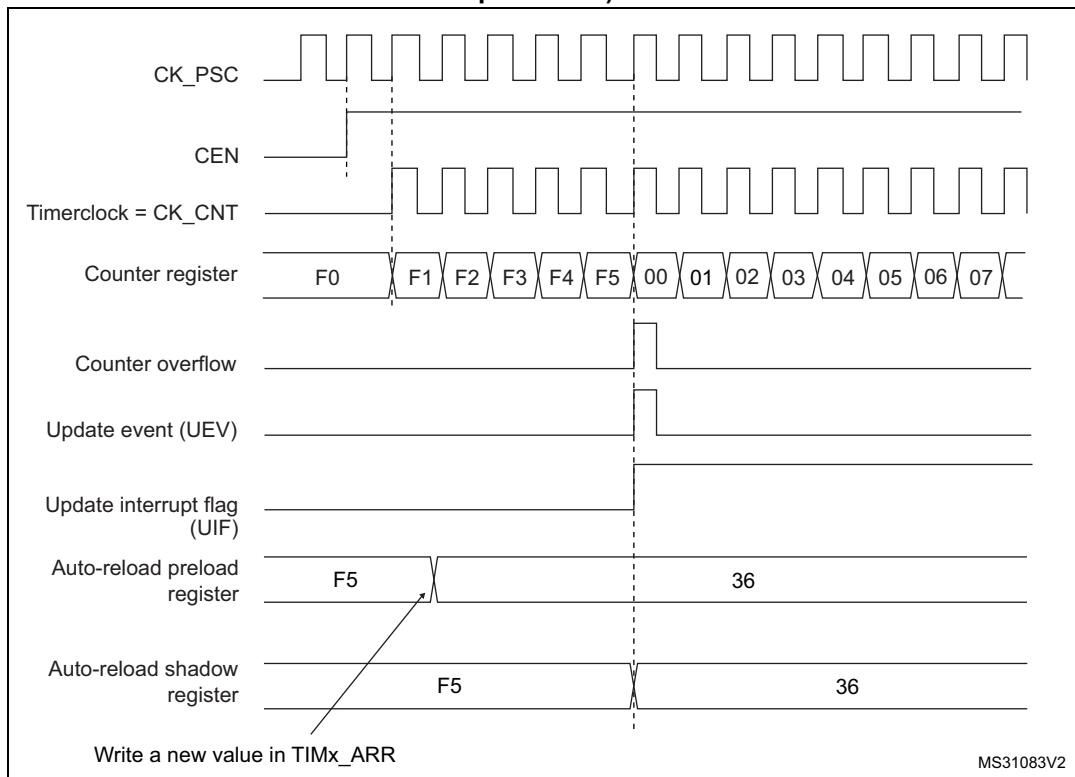
**Figure 118. Counter timing diagram, internal clock divided by 1**



**Figure 119. Counter timing diagram, internal clock divided by 2****Figure 120. Counter timing diagram, internal clock divided by 4**

**Figure 121. Counter timing diagram, internal clock divided by N****Figure 122. Counter timing diagram, Update event when ARPE=0 (TIMx\_ARR not preloaded)**

**Figure 123. Counter timing diagram, Update event when ARPE=1 (TIMx\_ARR preloaded)**



MS31083V2

### 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 generated 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).

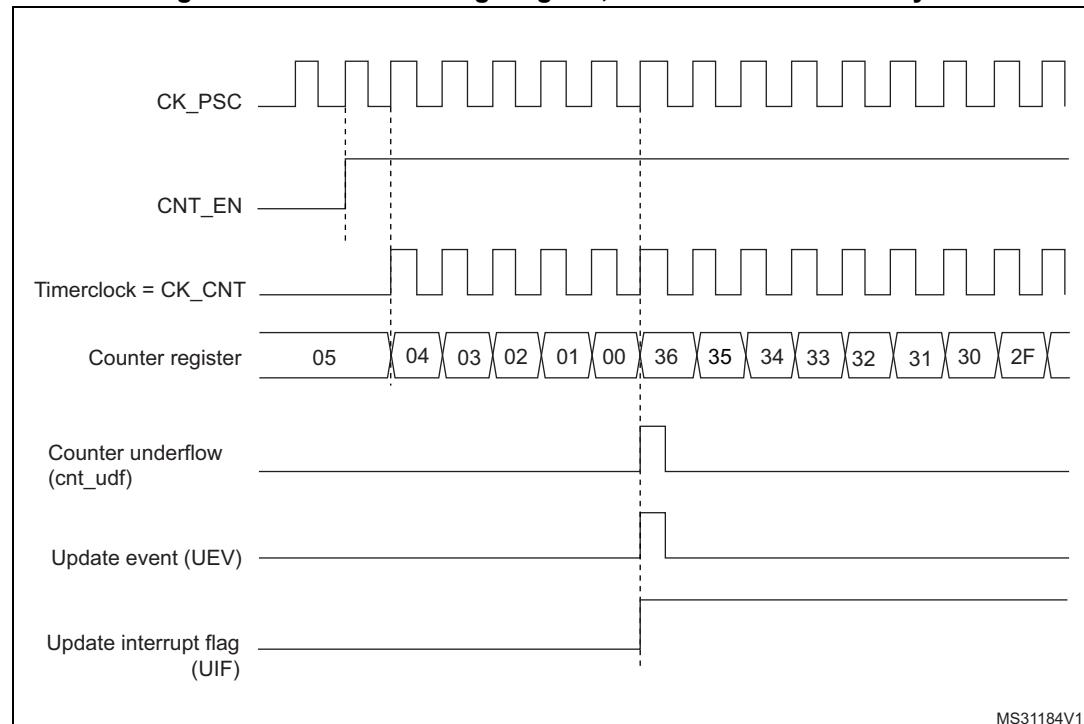
In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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):

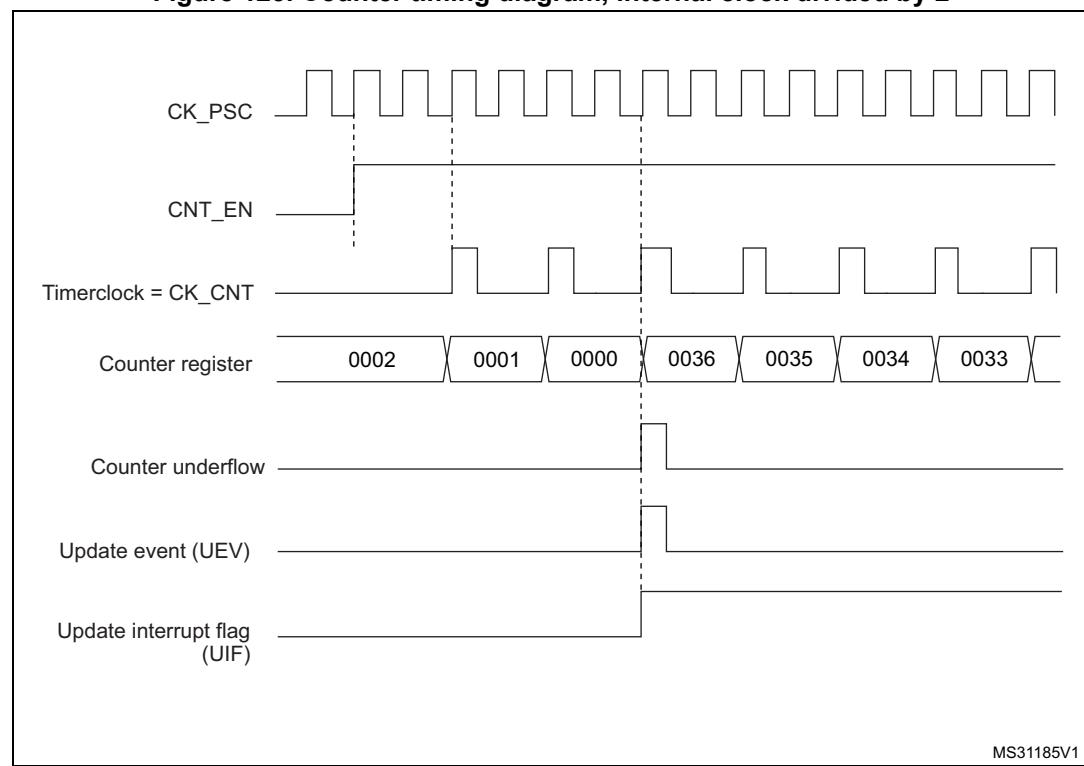
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 124. Counter timing diagram, internal clock divided by 1**



**Figure 125. Counter timing diagram, internal clock divided by 2**



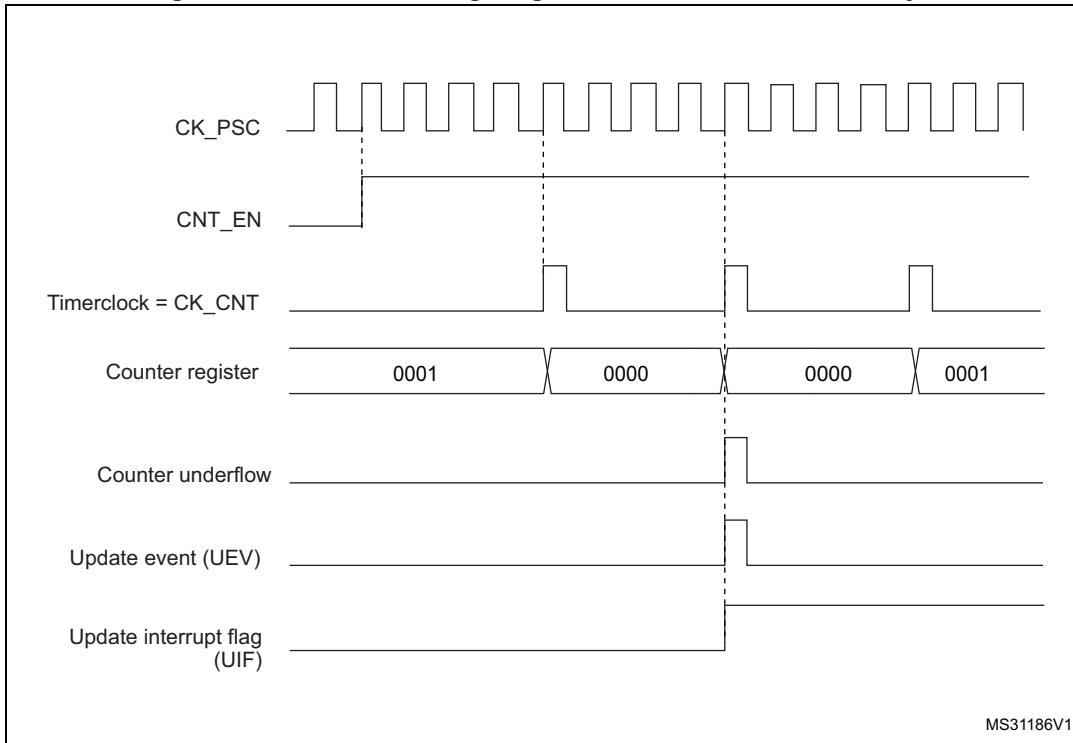
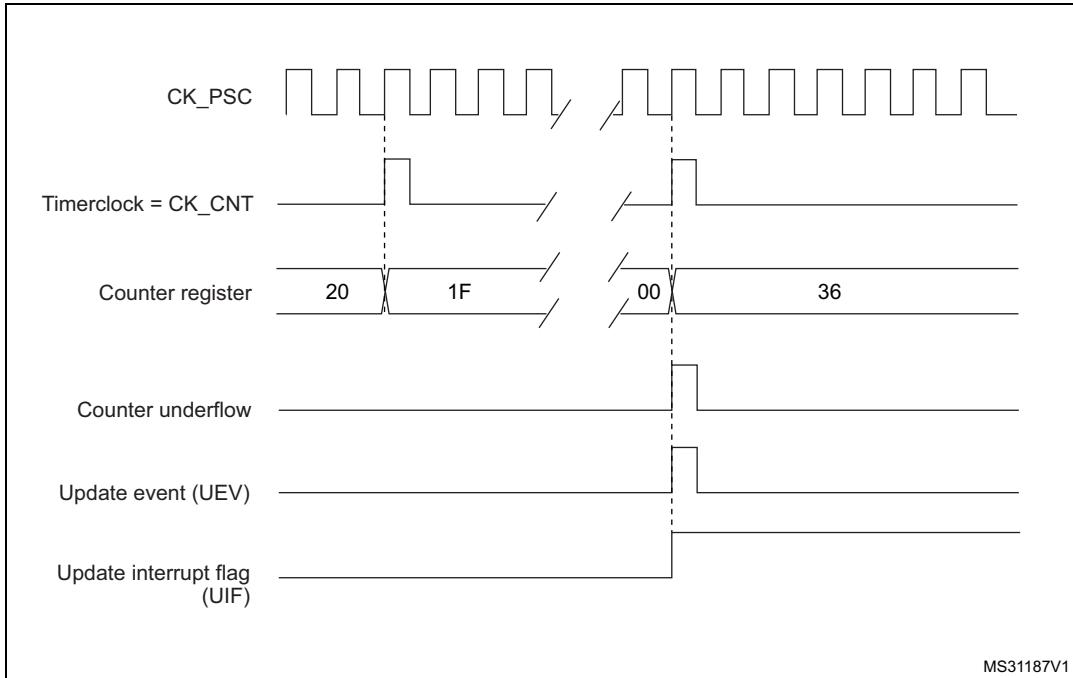
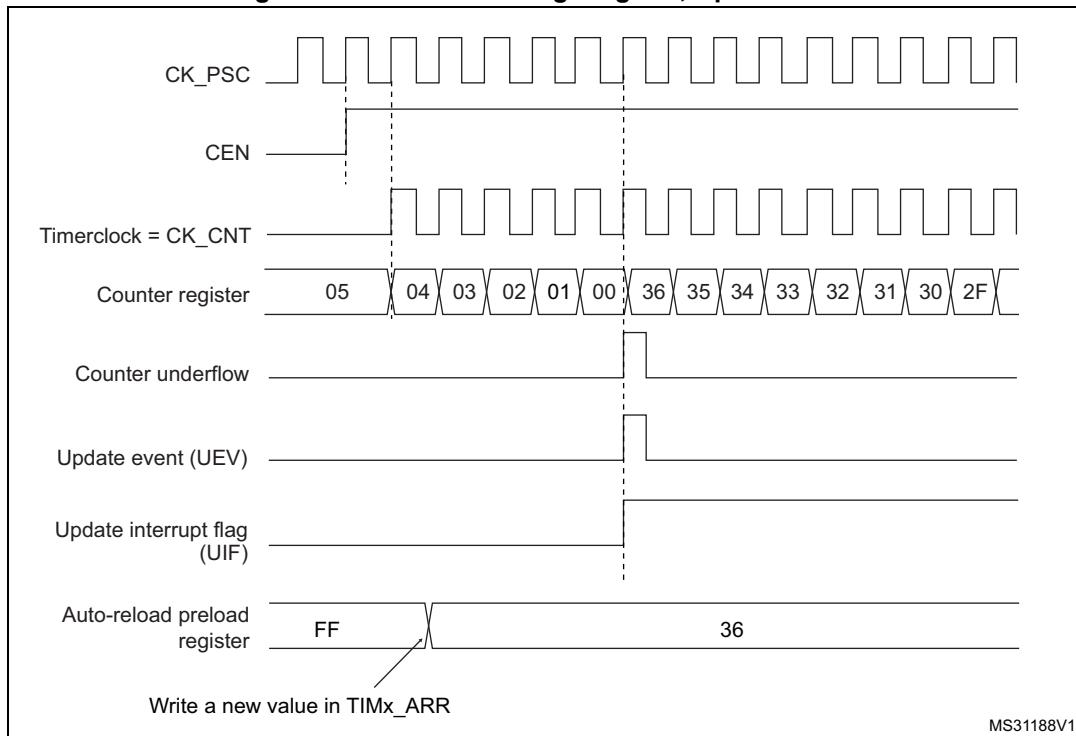
**Figure 126. Counter timing diagram, internal clock divided by 4****Figure 127. Counter timing diagram, internal clock divided by N**

Figure 128. Counter timing diagram, Update event



MS31188V1

### 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") or 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

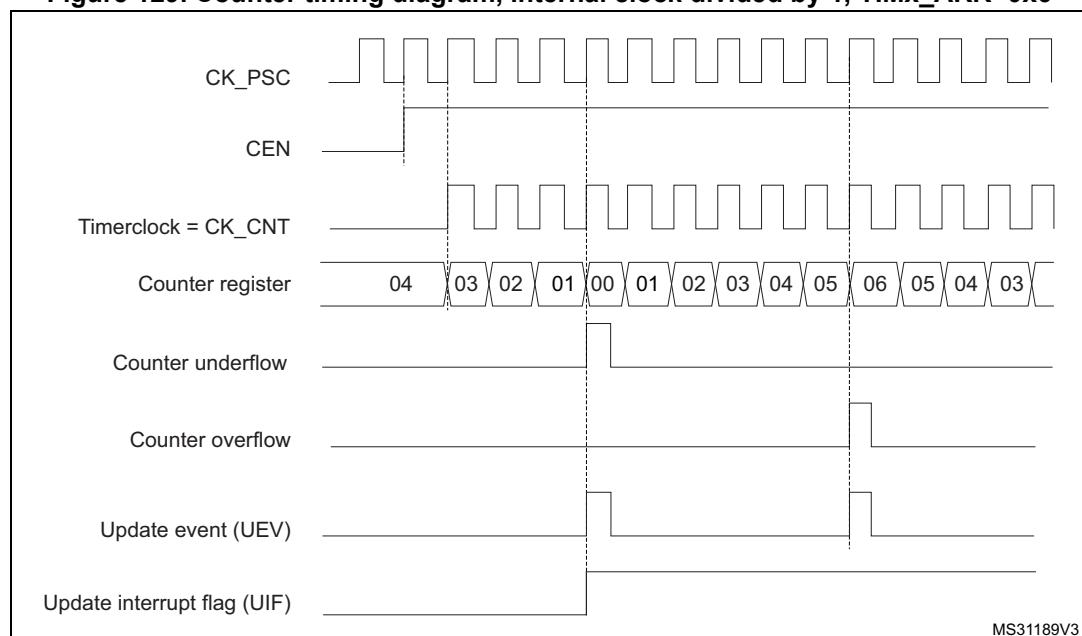
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

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):

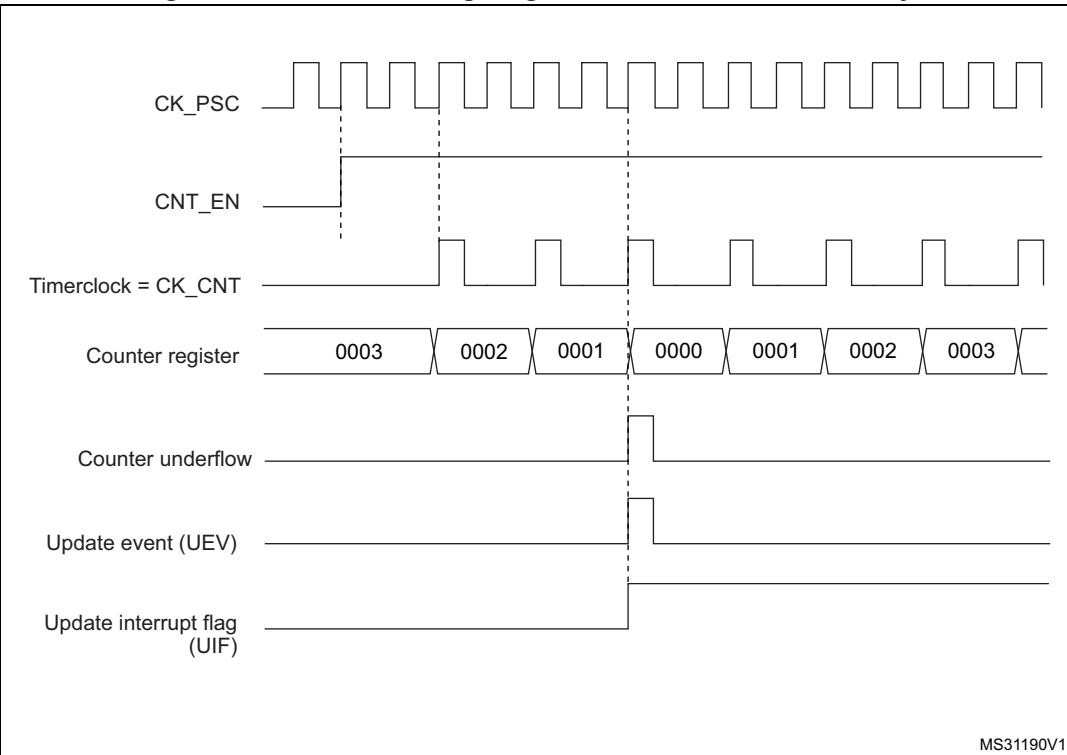
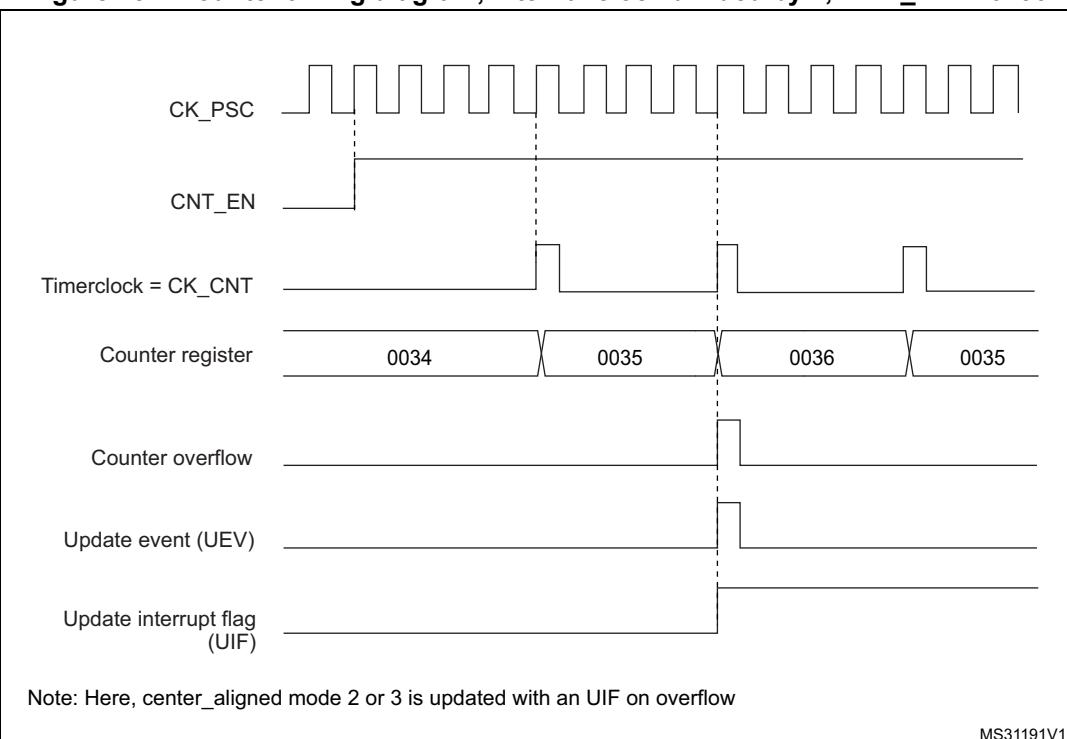
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter 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.

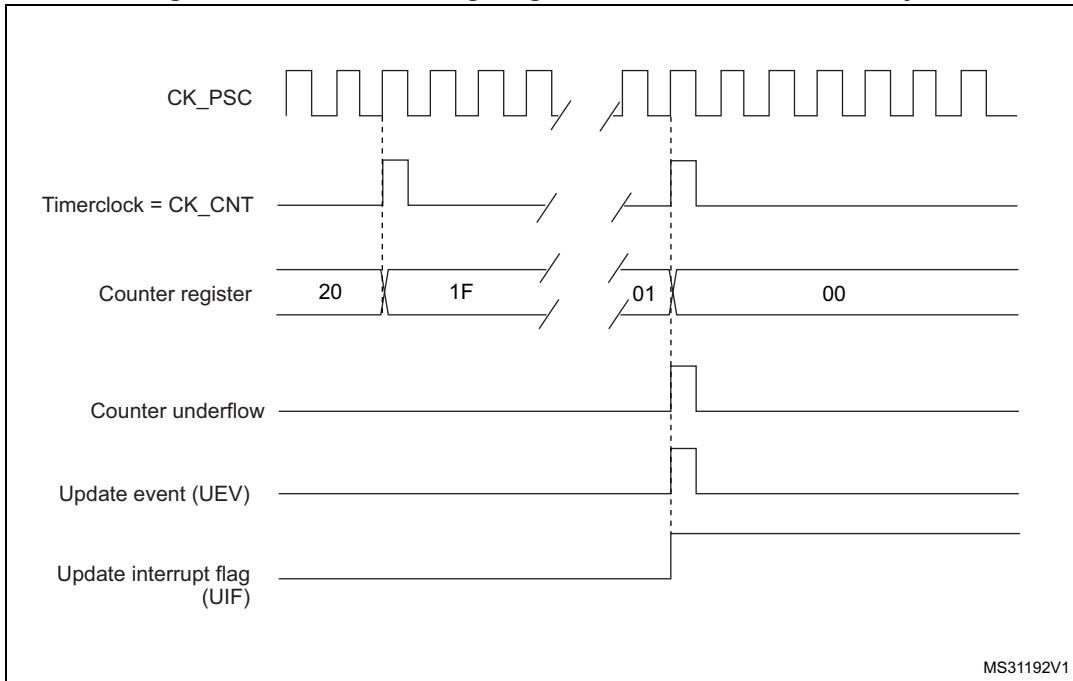
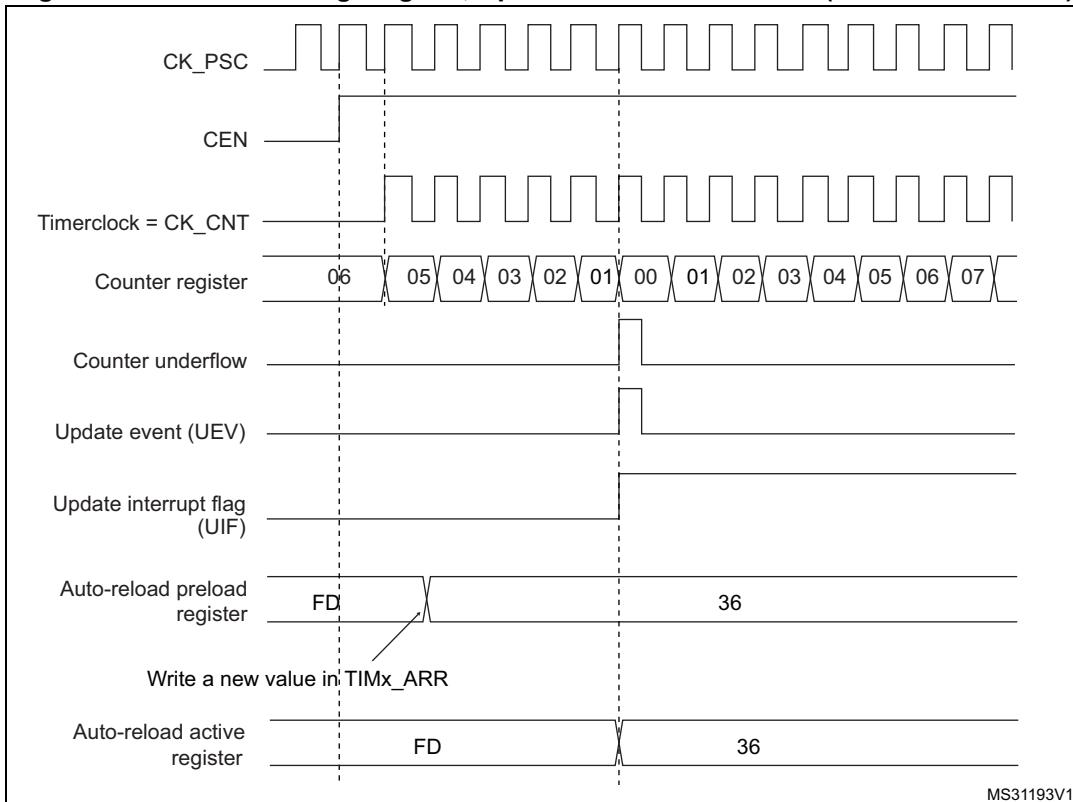
**Figure 129. Counter timing diagram, internal clock divided by 1, TIMx\_ARR=0x6**

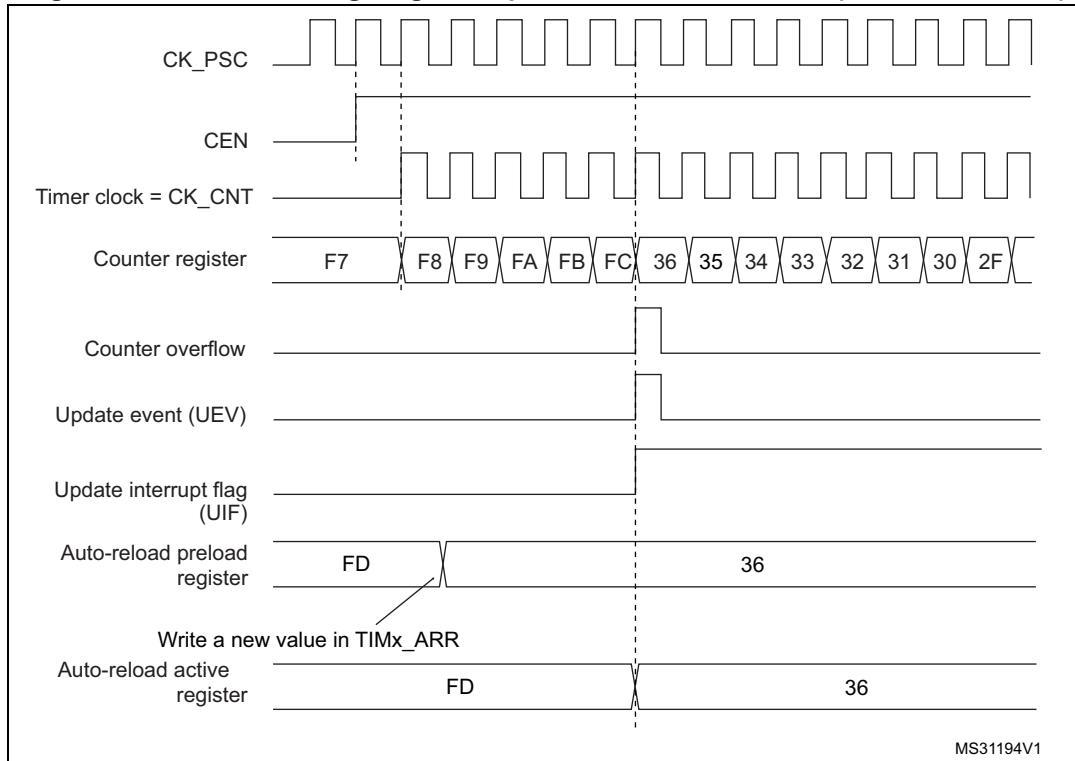


1. Here, center-aligned mode 1 is used (for more details refer to [Section 16.4.1: TIM3 control register 1 \(TIM3\\_CR1\) on page 441](#)).

**Figure 130. Counter timing diagram, internal clock divided by 2****Figure 131. 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 132. Counter timing diagram, internal clock divided by N****Figure 133. Counter timing diagram, Update event with ARPE=1 (counter underflow)**

**Figure 134. Counter timing diagram, Update event with ARPE=1 (counter overflow)**

### 16.3.3 Clock selection

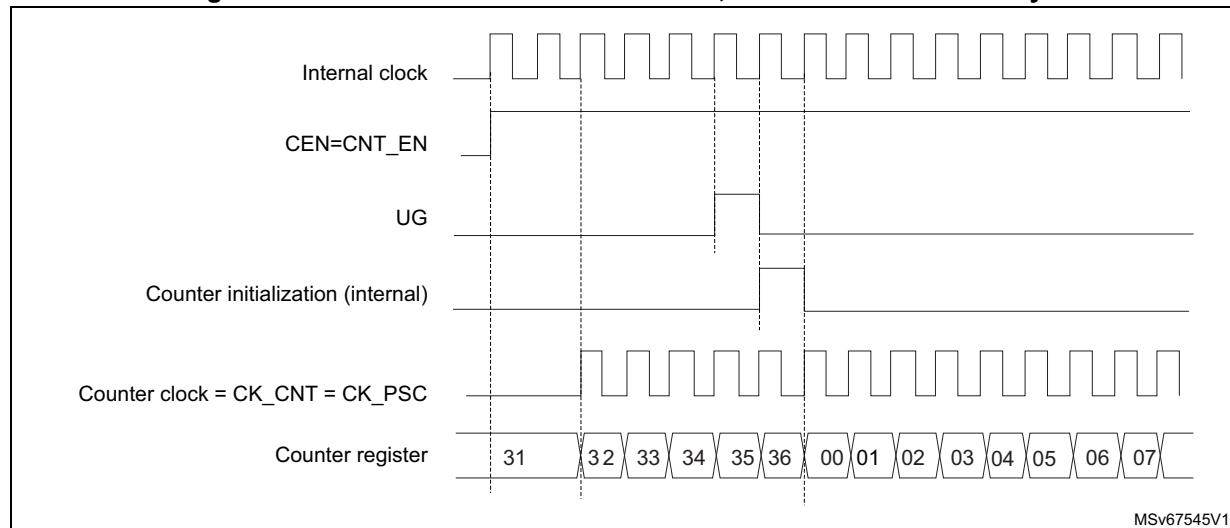
The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT)
- External clock mode1: external input pin (TI<sub>x</sub>)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to : [Using one timer as prescaler for another timer on page 435](#) for more details.

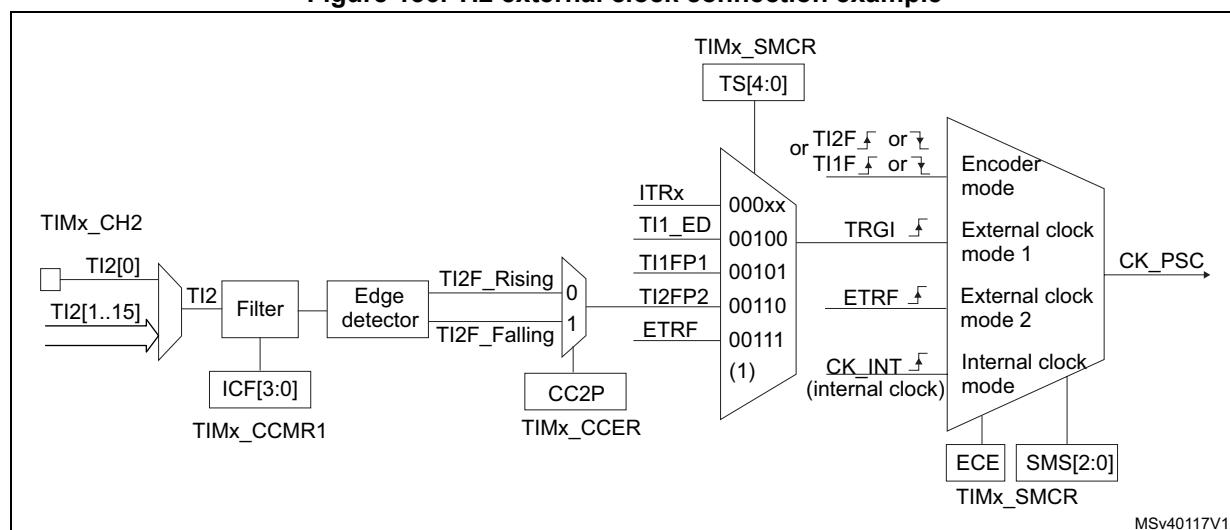
#### Internal clock source (CK\_INT)

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 135](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

**Figure 135. 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 136. TI2 external clock connection example**

1. Codes ranging from 01000 to 11111: ITRy.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= '01 in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).

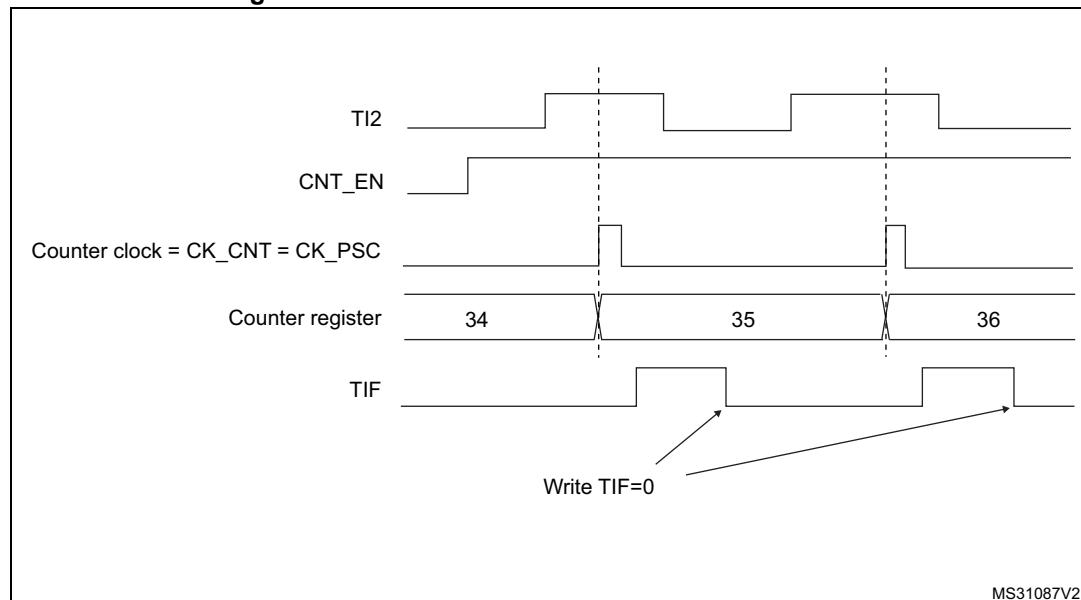
**Note:** The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx\_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
6. Select TI2 as the input source by writing TS=00110 in the TIMx\_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 137. Control circuit in external clock mode 1**



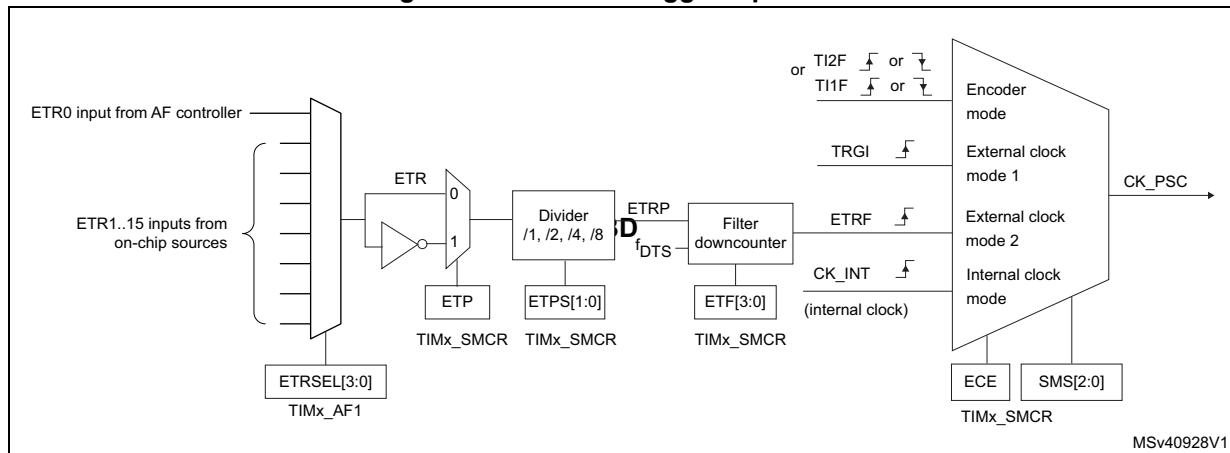
### External clock source mode 2

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 138](#) gives an overview of the external trigger input block.

Figure 138. External trigger input block



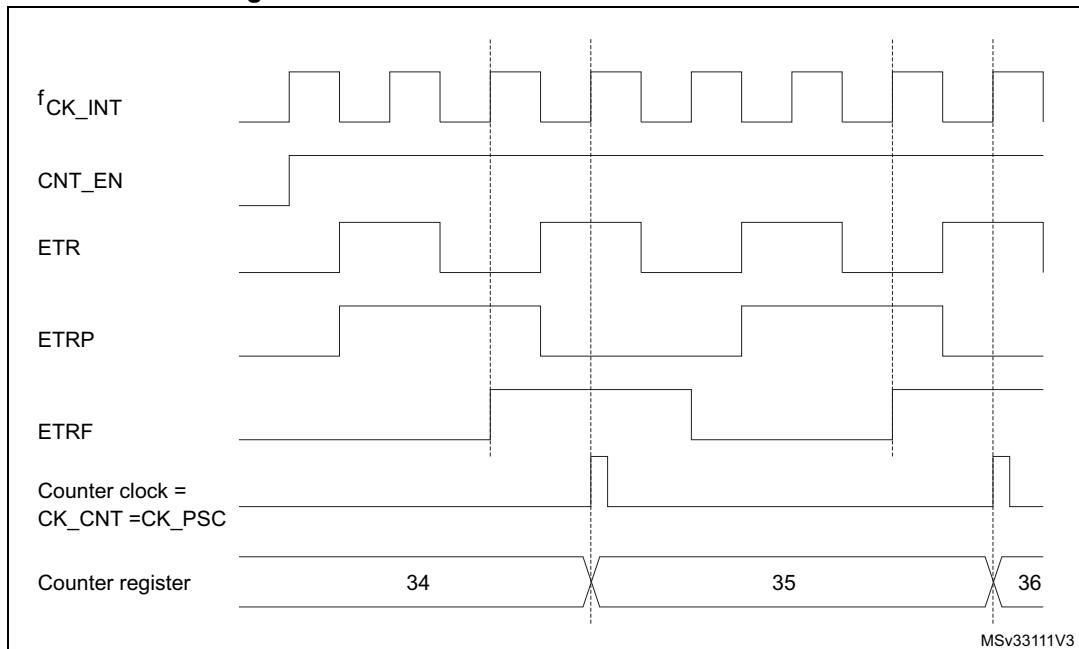
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx\_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx\_SMCR register.
3. Set the prescaler by writing ETSPS[1:0]=01 in the TIMx\_SMCR register.
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx\_SMCR register.
5. Enable external clock mode 2 by writing ECE=1 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most  $\frac{1}{4}$  of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETSPS prescaler setting.

Figure 139. Control circuit in external clock mode 2



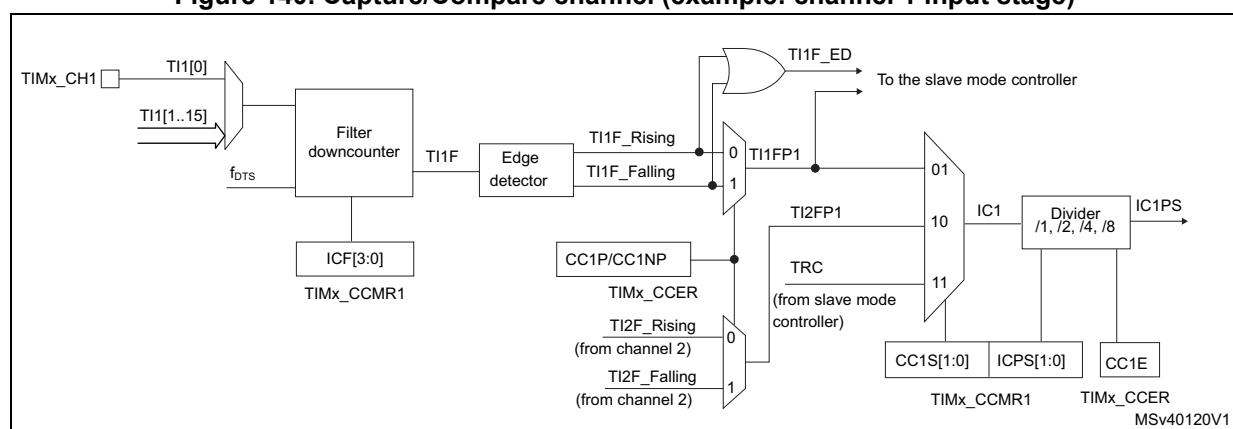
### 16.3.4 Capture/Compare channels

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 140. 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 141. Capture/Compare channel 1 main circuit

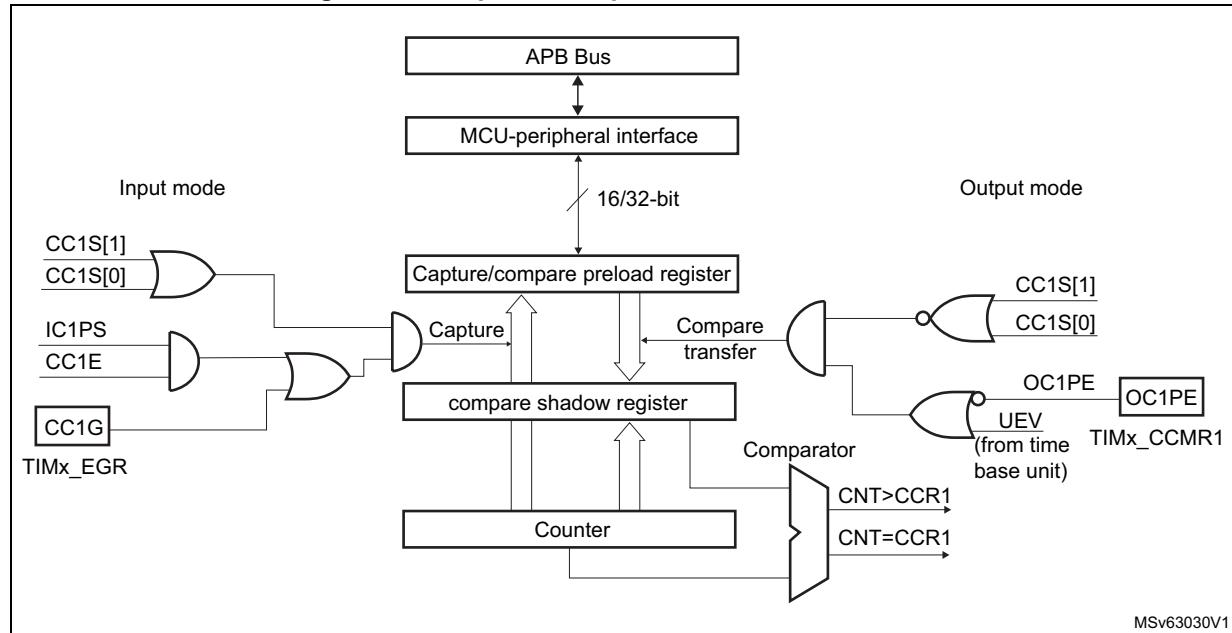
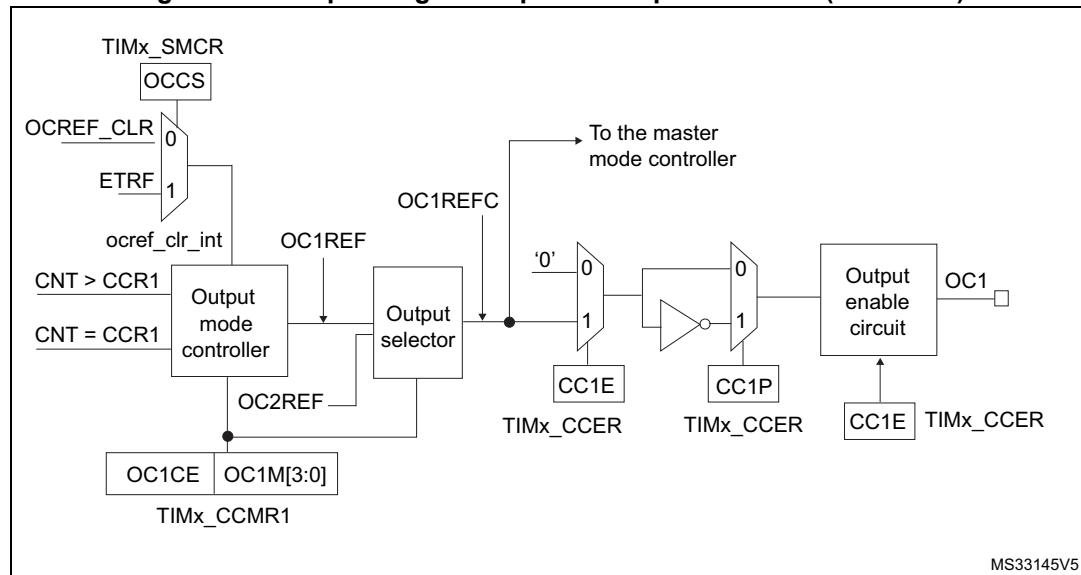


Figure 142. Output stage of Capture/Compare channel (channel 1)



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 IC<sub>x</sub> signal. When a capture occurs, the corresponding CC<sub>x</sub>IF 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 CC<sub>x</sub>IF flag was already high, then the over-capture flag CC<sub>x</sub>OF (TIMx\_SR register) is set. CC<sub>x</sub>IF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx\_CCRx register. CC<sub>x</sub>OF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TI<sub>x</sub> (IC<sub>x</sub>F 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<sub>DTS</sub> frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

**Note:** *IC interrupt and/or DMA requests can be generated by software by setting the corresponding CC<sub>x</sub>G bit in the TIMx\_EGR register.*

### 16.3.6 PWM input mode

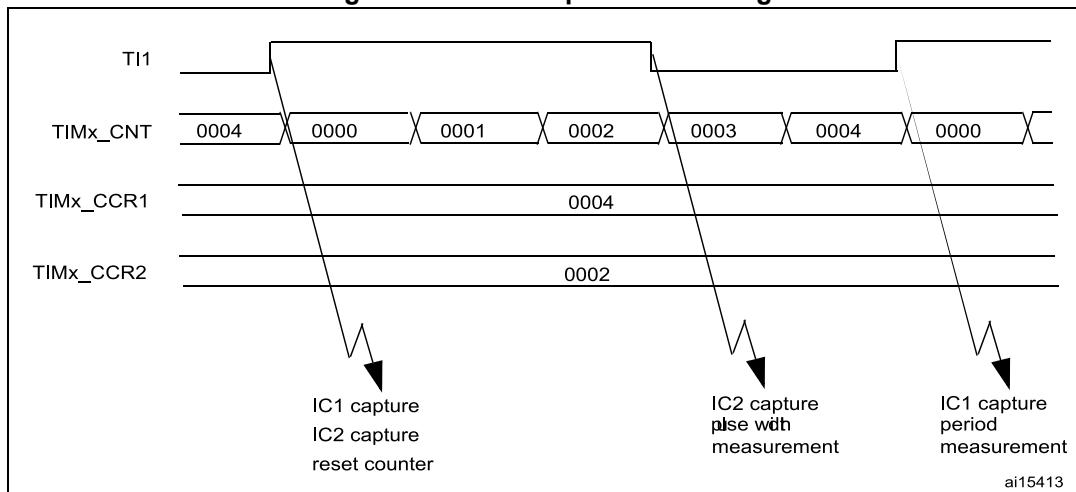
This mode is a particular case of input capture mode. The procedure is the same except:

- 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, one can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
4. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx\_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx\_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

**Figure 143. 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.

### 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, one just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

### 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.

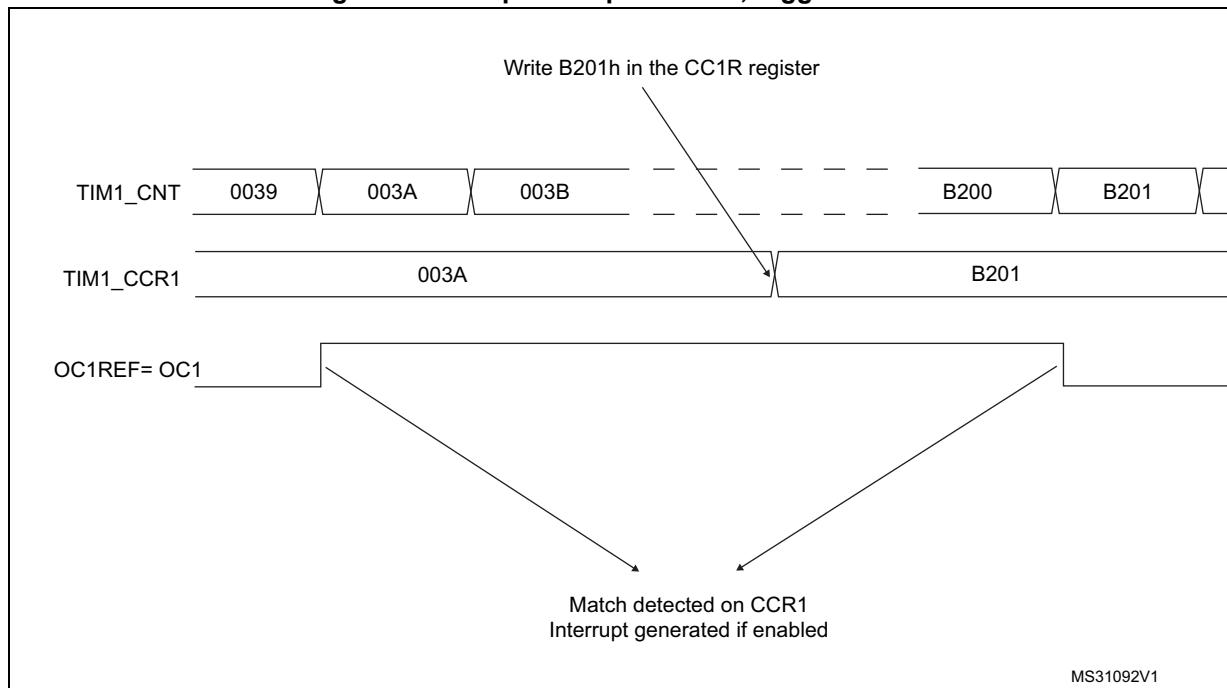
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

#### 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, one must write OCXM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 144](#).

**Figure 144. Output compare mode, toggle on OC1**



### 16.3.9 PWM mode

Pulse width modulation mode permits to generate a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or '111 (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_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 or
- When the output compare mode (OCxM bits in TIMx\_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM='000) to one of the PWM modes (OCxM='110 or '111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx\_CR1 register.

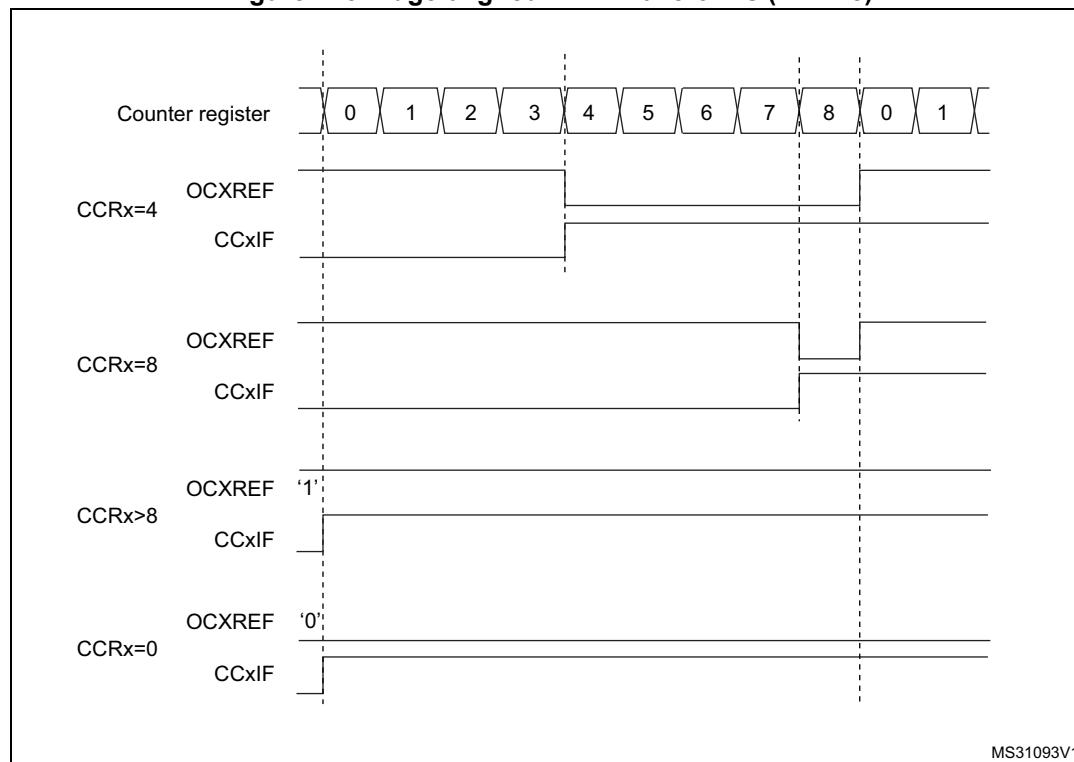
### PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to [Upcounting mode on page 400](#).

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 145](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 145. Edge-aligned PWM waveforms (ARR=8)**



MS31093V1

## Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to [Downcounting mode on page 403](#).

In PWM mode 1, the reference signal ocxref is low as long as TIMx\_CNT>TIMx\_CCRx else it becomes high. If the compare value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

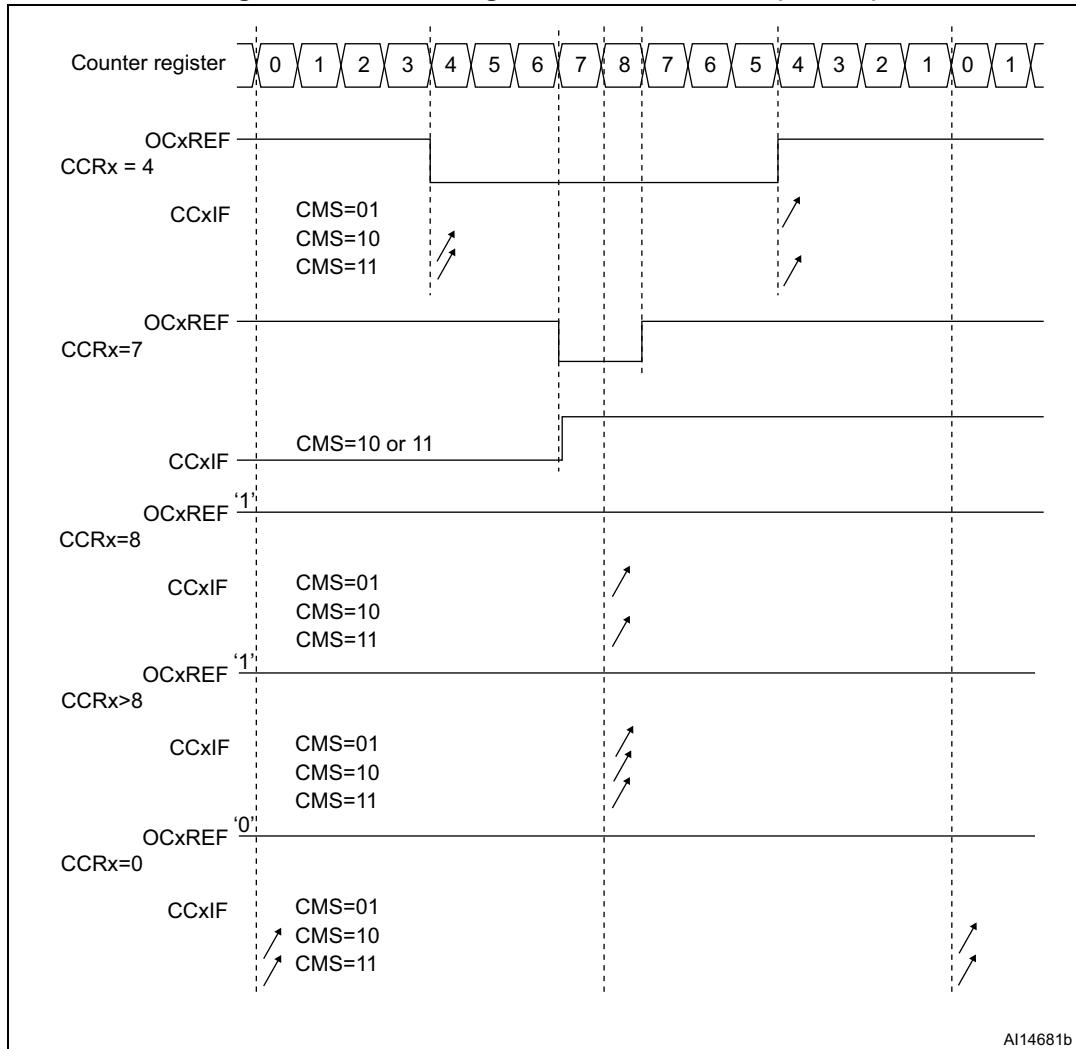
## PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to [Center-aligned mode \(up/down counting\) on page 406](#).

*Figure 146* 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 146. Center-aligned PWM waveforms (ARR=8)



AI14681b

**Hints on using center-aligned mode:**

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx\_CNT>TIMx\_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx\_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

### 16.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx\_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

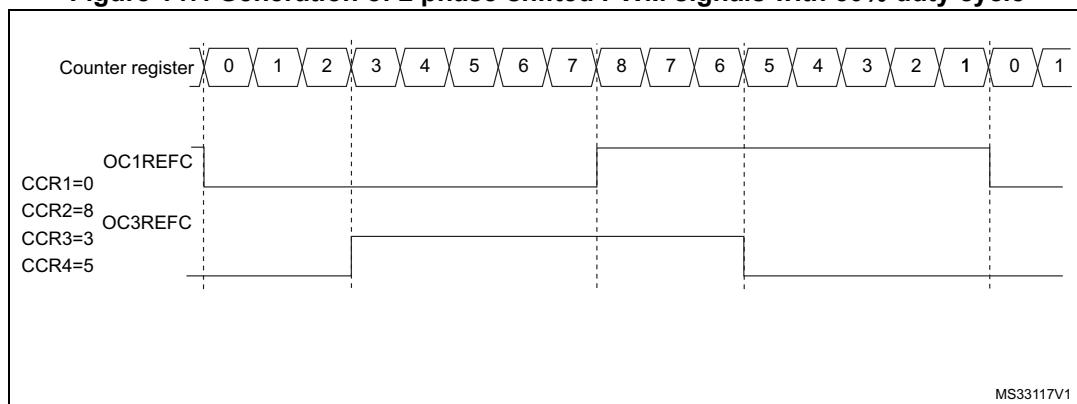
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

**Note:** *The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

*Figure 147* shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

**Figure 147. Generation of 2 phase-shifted PWM signals with 50% duty cycle**



### 16.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx\_ARR register, the duty cycle and delay are determined by the two TIMx\_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMS:

- OC1REFC (or OC2REFC) is controlled by TIMx\_CCR1 and TIMx\_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx\_CCR3 and TIMx\_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

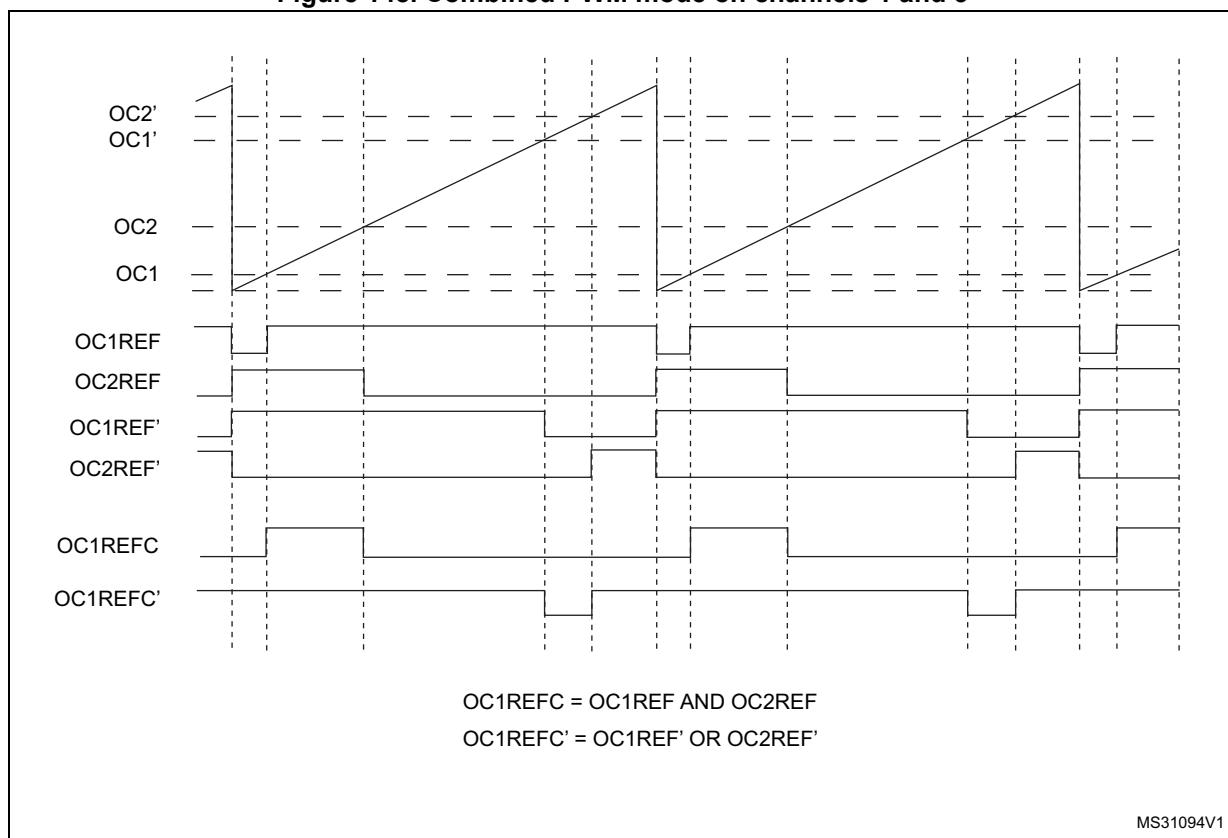
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** *The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

*Figure 148* shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

**Figure 148. Combined PWM mode on channels 1 and 3**



### 16.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref\_clr\_int input (OCxCE enable bit in the corresponding TIMx\_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

OCREF\_CLR\_INPUT can be selected between the OCREF\_CLR input and ETRF (ETR after the filter) by configuring the OCCS bit in the TIMx\_SMCR register.

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx\_CCMRx register). OCxREF remains low until the next transition to the active state, on the following PWM cycle.

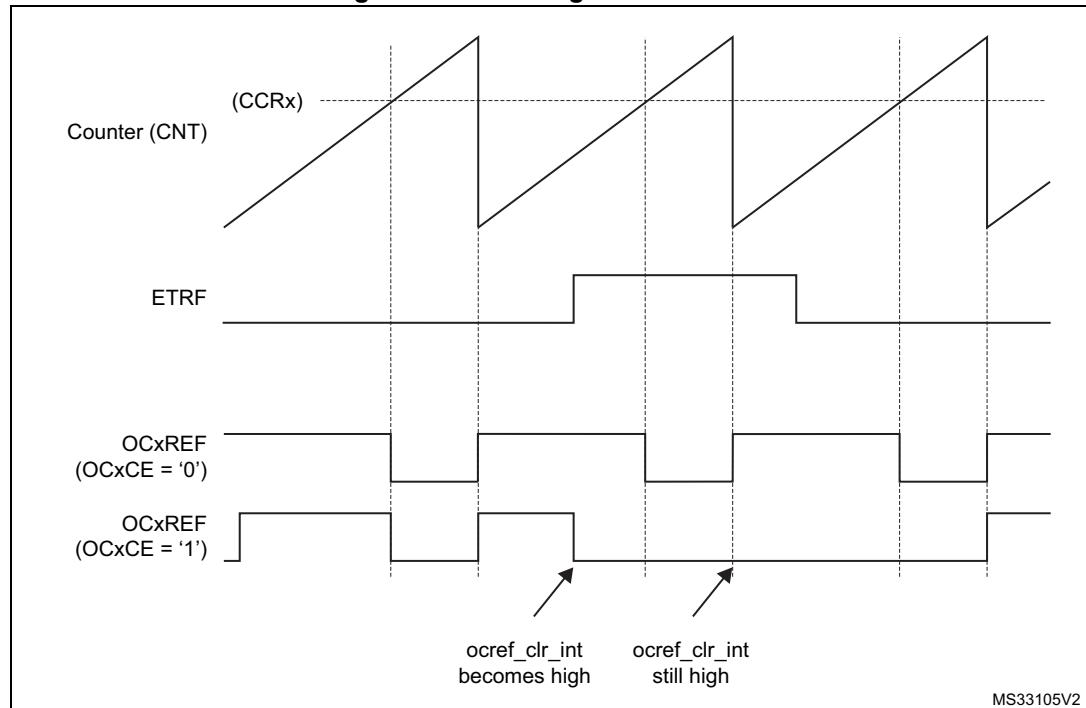
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx\_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1\_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

*Figure 149* 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 149. Clearing TIMx OCxREF**



**Note:**

*In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.*

### 16.3.13 One-pulse mode

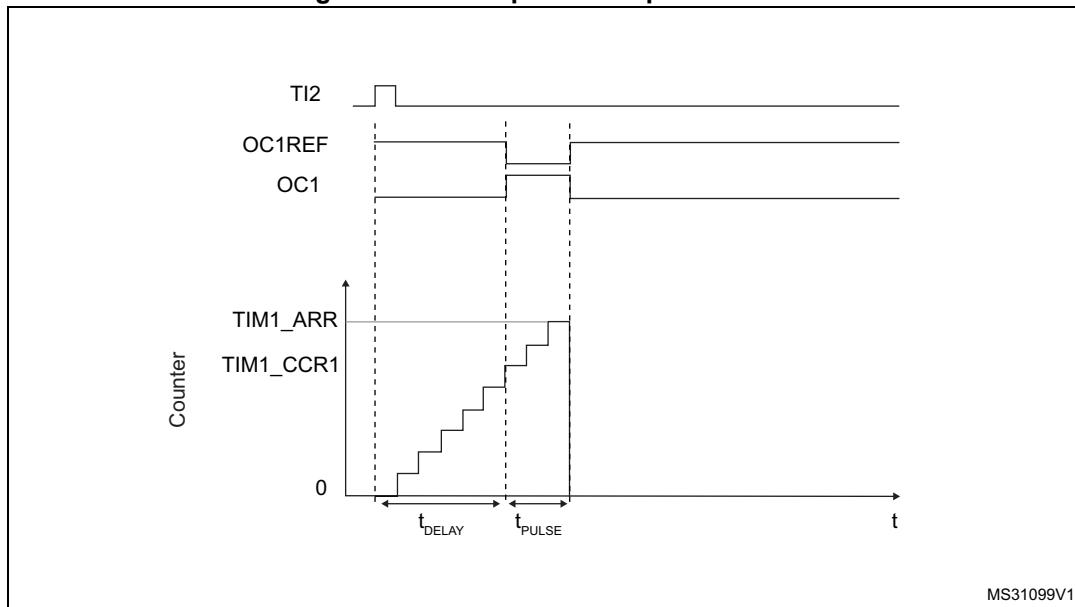
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCR<sub>x</sub> ≤ ARR (in particular, 0<CCR<sub>x</sub>),

**Figure 150. Example of one-pulse mode.**



For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx\_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx\_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx\_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  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 one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx\_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case one has to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx\_CR1 register is set to '0', so the Repetitive Mode is selected.

#### **Particular case: OCx fast enable:**

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx\_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### **16.3.14 Retriggerable one pulse mode**

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 16.3.13](#):

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx\_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

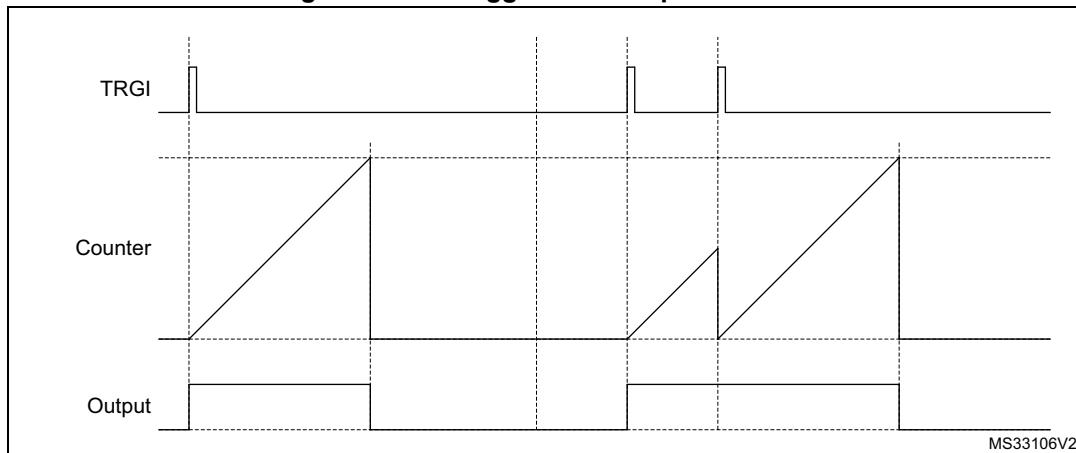
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

*Note:* In retriggerable one pulse mode, the CCxIF flag is not significant.

*The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.*

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx\_CR1.*

Figure 151. Retriggerable one-pulse mode.



### 16.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx\_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx\_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to [Table 68](#). The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx\_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx\_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx\_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx\_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the-quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

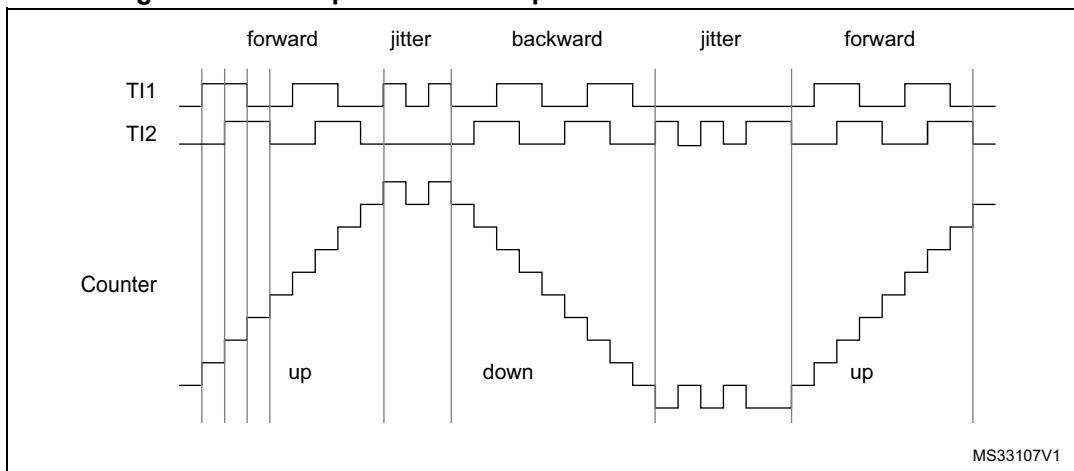
**Table 68. Counting direction versus encoder signals**

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

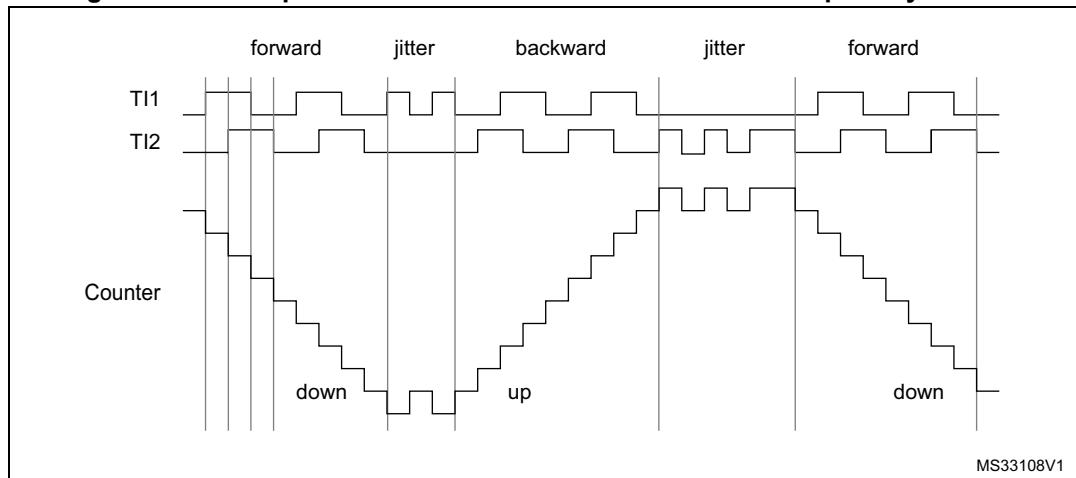
An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

*Figure 152* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx\_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx\_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx\_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx\_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx\_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx\_CR1 register, Counter is enabled)

**Figure 152. Example of counter operation in encoder interface mode**

*Figure 153* gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

**Figure 153. Example of encoder interface mode with TI1FP1 polarity inverted**

The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

### 16.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

### 16.3.17 Timer input XOR function

The TI1S bit in the TIM1xx\_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx\_CH1 to TIMx\_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in [Section 15.3.25: Interfacing with Hall sensors on page 349](#).

### 16.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

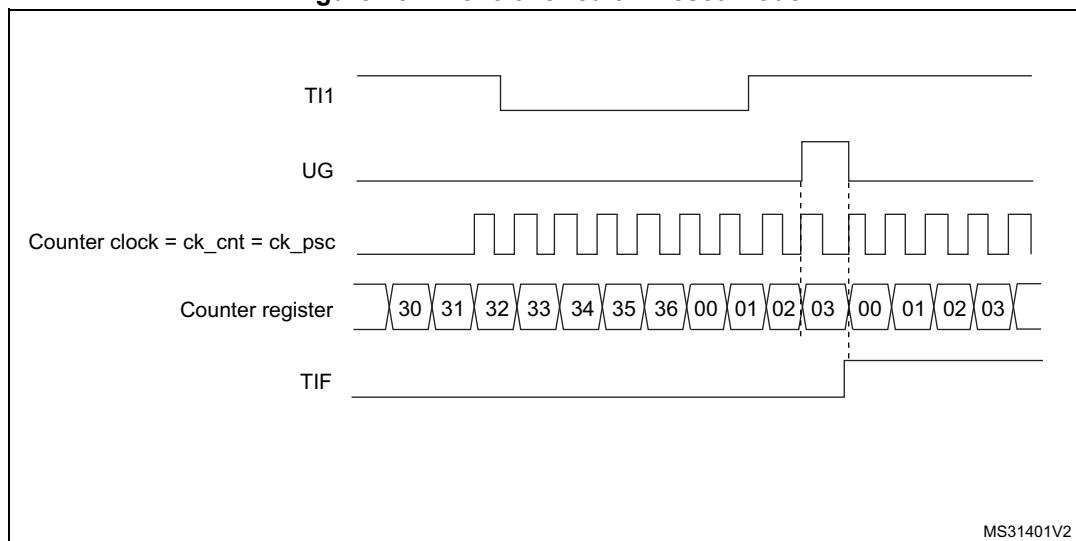
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

**Figure 154. Control circuit in reset mode**



### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

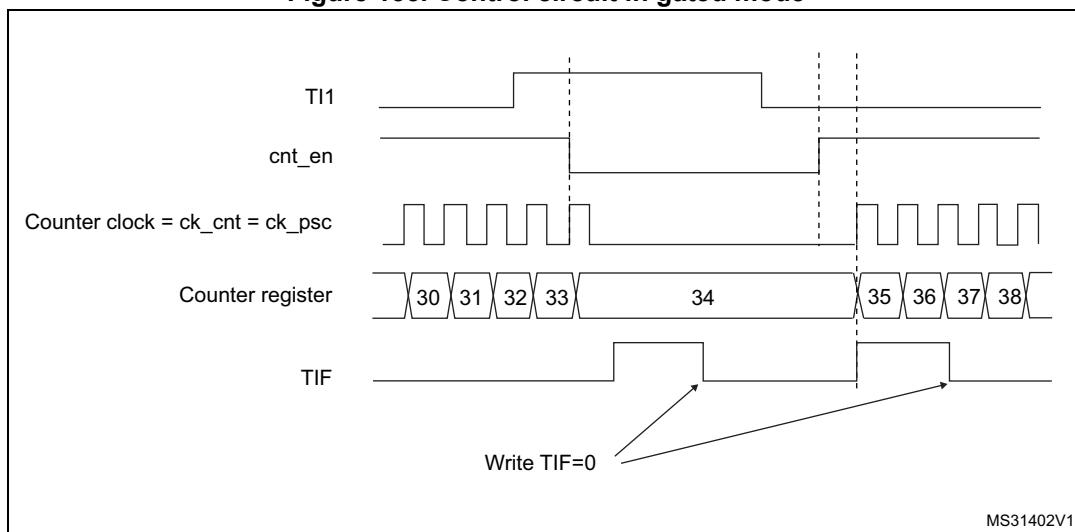
1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write

- CC1P=1 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
  3. Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 155. 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.

**Note:**

*The configuration “CCxP=CCxNP=1” (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.*

### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

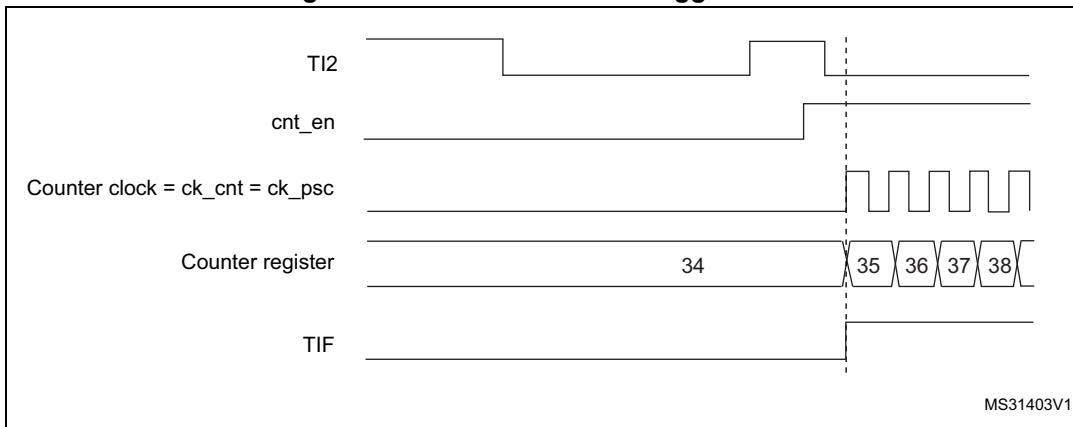
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx\_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 156. Control circuit in trigger mode**



### Slave mode: External Clock mode 2 + trigger mode

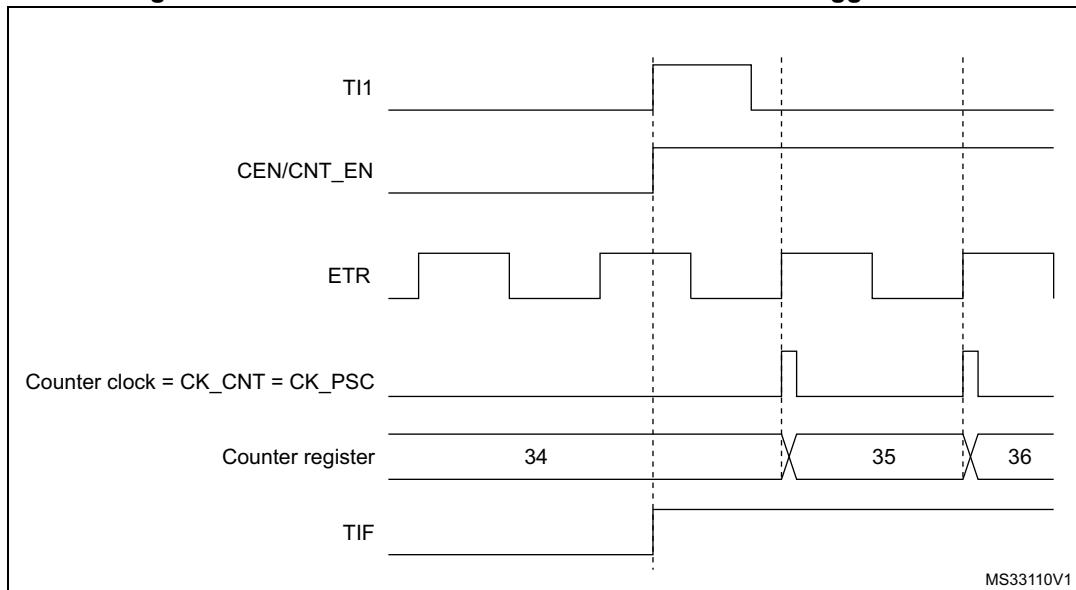
The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx\_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx\_SMCR register as follows:
  - ETF = 0000: no filter
  - ETPS=00: prescaler disabled
  - ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
  - IC1F=0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S=01 in TIMx\_CCMR1 register to select only the input capture source
  - CC1P=0 and CC1NP=0 in TIMx\_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

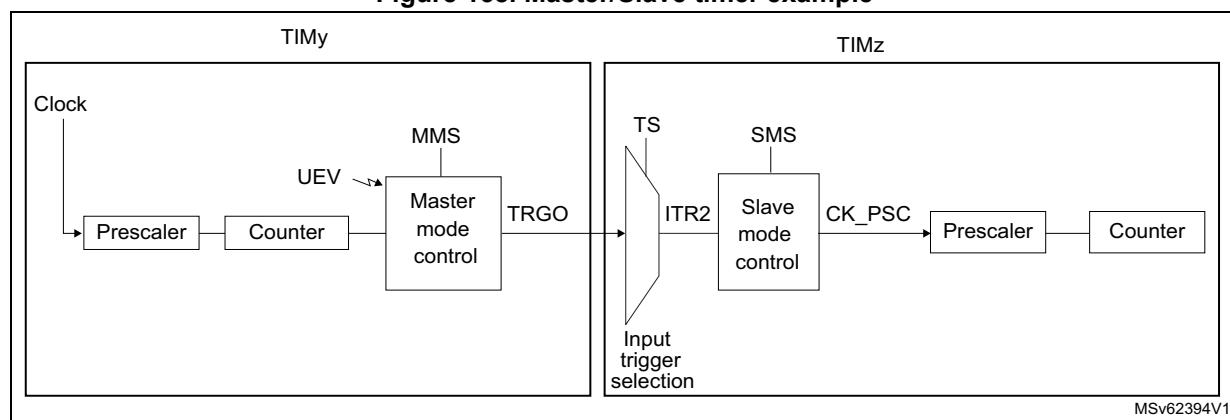
The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

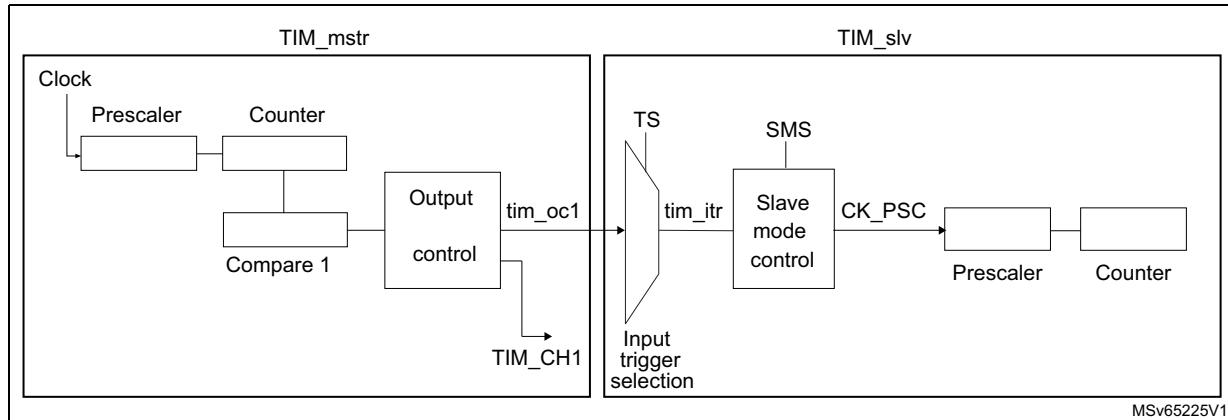
**Figure 157. Control circuit in external clock mode 2 + trigger mode**

### 16.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

*Figure 158: Master/Slave timer example* and *Figure 159: Master/slave connection example with 1 channel only timers* present an overview of the trigger selection and the master mode selection blocks.

**Figure 158. Master/Slave timer example**

**Figure 159. Master/slave connection example with 1 channel only timers**

**Note:** The timers with one channel only (see [Figure 159](#)) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx\_SMCR register on the device to identify which timers can be targeted as slave. The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination’s timer CK\_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

### Using one timer as prescaler for another timer

For example, TIMy can be configured to act as a prescaler for TIMz. Refer to [Figure 158](#). To do this:

1. Configure TIMy in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIMy\_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIMy to TIMz, TIMz must be configured in slave mode using ITR as internal trigger. This is selected through the TS bits in the TIMz\_SMCR register (writing TS=00).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIMz\_SMCR register). This causes TIMz to be clocked by the rising edge of the periodic TIMy trigger signal (which correspond to the TIMy counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx\_CR1 register).

**Note:** If OCx is selected on TIMy as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIMz.

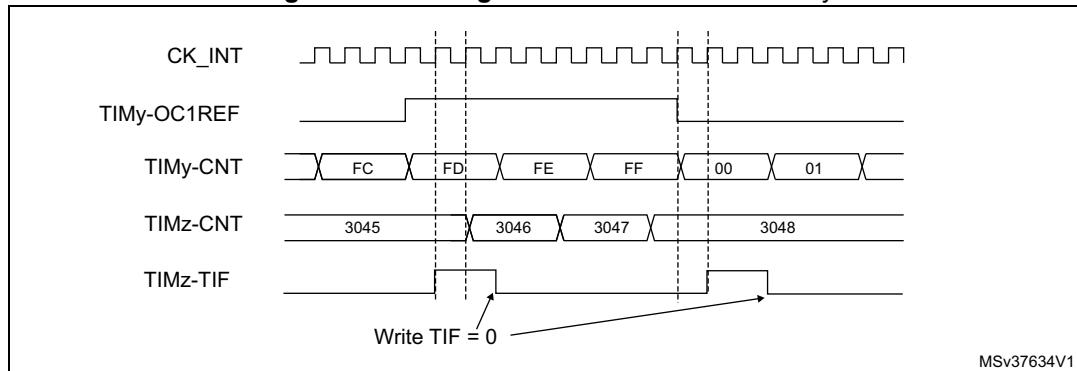
### Using one timer to enable another timer

In this example, we control the enable of TIMz with the output compare 1 of Timer y. Refer to [Figure 158](#) for connections. TIMz counts on the divided internal clock only when OC1REF of TIMy 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 TIMy master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIMy\_CR2 register).
2. Configure the TIMy OC1REF waveform (TIMy\_CCMR1 register).
3. Configure TIMz to get the input trigger from TIMy (TS=00 in the TIMz\_SMCR register).
4. Configure TIMz in gated mode (SMS=101 in TIMz\_SMCR register).
5. Enable TIMz by writing '1 in the CEN bit (TIMz\_CR1 register).
6. Start TIMy by writing '1 in the CEN bit (TIMy\_CR1 register).

**Note:** *The counter z clock is not synchronized with counter 1, this mode only affects the TIMz counter enable signal.*

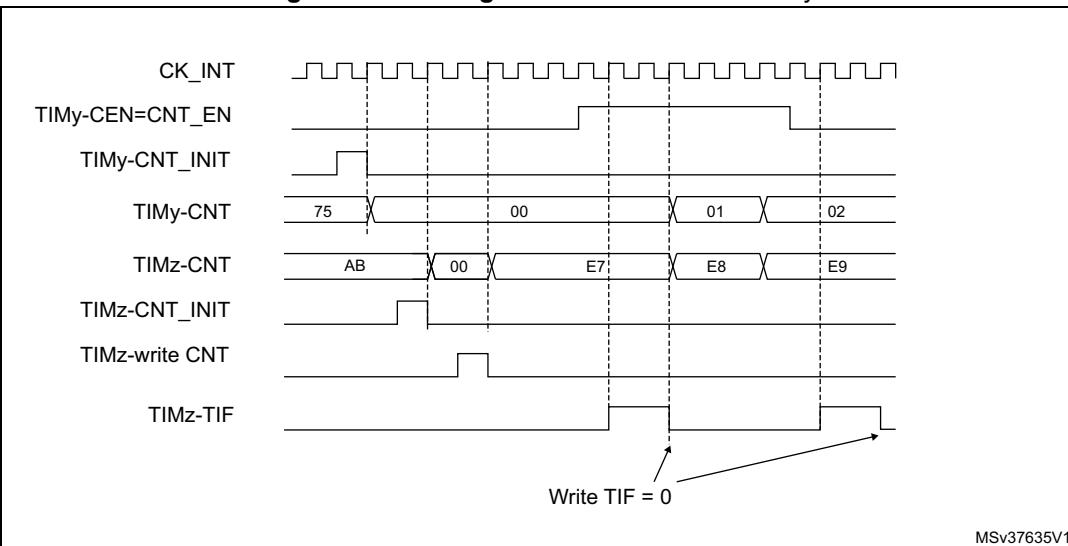
**Figure 160. Gating TIMz with OC1REF of TIMy**



In the example in [Figure 160](#), the TIMz 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 TIMy. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx\_EGR registers.

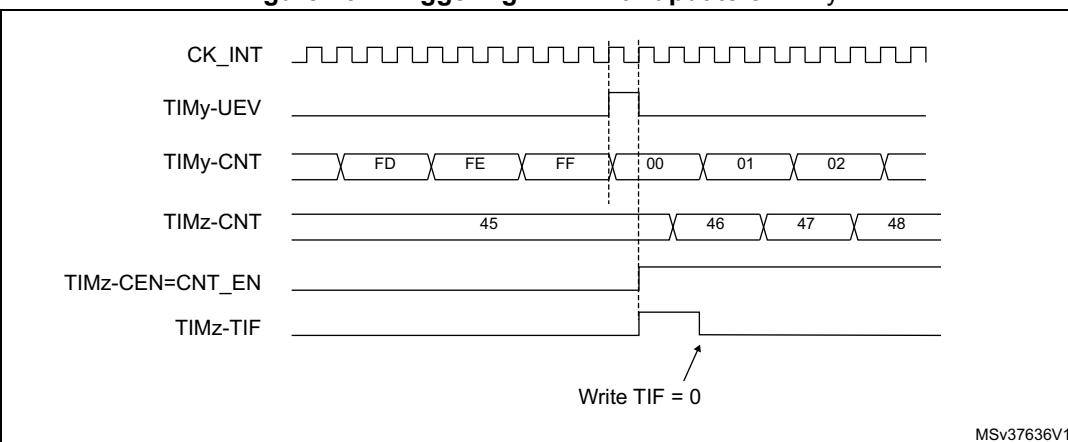
In the next example (refer to [Figure 161](#)), we synchronize TIMy and TIMz. TIMy is the master and starts from 0. TIMz is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIMz stops when TIMy is disabled by writing '0 to the CEN bit in the TIMy\_CR1 register:

1. Configure TIMy master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIMy\_CR2 register).
2. Configure the TIMy OC1REF waveform (TIMy\_CCMR1 register).
3. Configure TIMz to get the input trigger from TIMy (TS=00 in the TIMz\_SMCR register).
4. Configure TIMz in gated mode (SMS=101 in TIMz\_SMCR register).
5. Reset TIMy by writing '1 in UG bit (TIMy\_EGR register).
6. Reset TIMz by writing '1 in UG bit (TIMz\_EGR register).
7. Initialize TIMz to 0xE7 by writing '0xE7' in the TIMz counter (TIMz\_CNTL).
8. Enable TIMz by writing '1 in the CEN bit (TIMz\_CR1 register).
9. Start TIMy by writing '1 in the CEN bit (TIMy\_CR1 register).
10. Stop TIMy by writing '0 in the CEN bit (TIMy\_CR1 register).

**Figure 161. Gating TIMz with Enable of TIMy****Using one timer to start another timer**

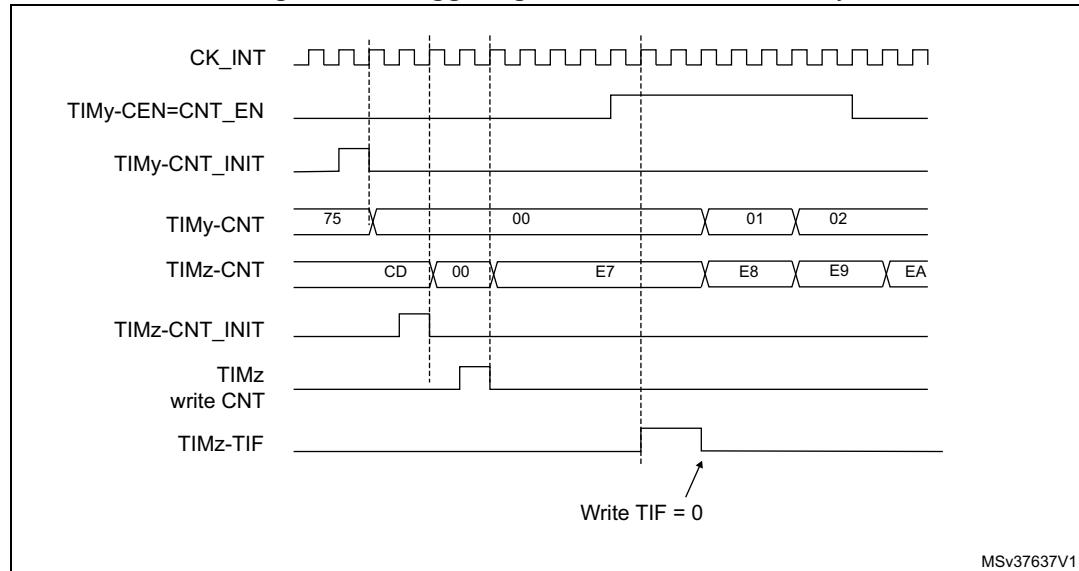
In this example, we set the enable of Timer z with the update event of Timer y. Refer to [Figure 158](#) for connections. Timer z starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer z receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0' to the CEN bit in the TIMz\_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 TIMy master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIMy\_CR2 register).
2. Configure the TIMy period (TIMy\_ARR registers).
3. Configure TIMz to get the input trigger from TIMy (TS=00 in the TIMz\_SMCR register).
4. Configure TIMz in trigger mode (SMS=110 in TIMz\_SMCR register).
5. Start TIMy by writing '1 in the CEN bit (TIMy\_CR1 register).

**Figure 162. Triggering TIMz with update of TIMy**

As in the previous example, both counters can be initialized before starting counting. [Figure 163](#) shows the behavior with the same configuration as in [Figure 162](#) but in trigger mode instead of gated mode (SMS=110 in the TIMz\_SMCR register).

**Figure 163. Triggering TIMz with Enable of TIMy**



### Starting 2 timers synchronously in response to an external trigger

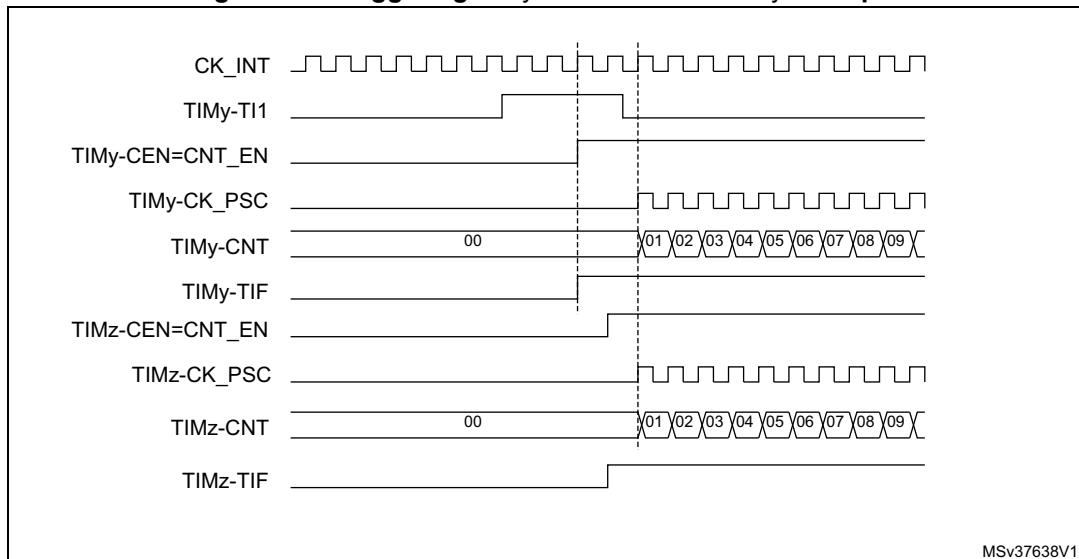
In this example, we set the enable of TIMy when its TI1 input rises, and the enable of TIMz with the enable of TIMy. Refer to [Figure 158](#) for connections. To ensure the counters are aligned, TIMy must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIMz):

1. Configure TIMy master mode to send its Enable as trigger output (MMS=001 in the TIMy\_CR2 register).
2. Configure TIMy slave mode to get the input trigger from TI1 (TS=00100 in the TIMy\_SMCR register).
3. Configure TIMy in trigger mode (SMS=110 in the TIMy\_SMCR register).
4. Configure the TIMy in Master/Slave mode by writing MSM=1 (TIMy\_SMCR register).
5. Configure TIMz to get the input trigger from TIMy (TS=00000 in the TIMz\_SMCR register).
6. Configure TIMz in trigger mode (SMS=110 in the TIMz\_SMCR register).

When a rising edge occurs on TI1 (TIMy), both counters starts counting synchronously on the internal clock and both TIF flags are set.

**Note:**

*In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx\_CNT). One can see that the master/slave mode insert a delay between CNT\_EN and CK\_PSC on TIMy.*

**Figure 164. Triggering TIMy and TIMz with TIMy TI1 input****Note:**

*The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO<sub>2</sub> signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

### 16.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx\_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx\_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx\_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx\_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register:

Example:

00000: TIMx\_CR1

00001: TIMx\_CR2

00010: TIMx\_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCR<sub>x</sub> registers ( $x = 2, 3, 4$ ) upon an update event, with the DMA transferring half words into the CCR<sub>x</sub> registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:  
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 16.3.21 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M0+ core - halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit inDBGMCU module. For more details, refer to [Section 26.9.2: Debug support for timers, watchdog, and I2C](#).

## 16.4 TIM3 registers

In this section, “TIMx” should be understood as “TIM3” since there is only one instance of this type of timer for the products to which this reference manual applies.

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

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

### 16.4.1 TIM3 control register 1 (TIM3\_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.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN	
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

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

Bit 11 **UIFREMAP**: UIF status bit remapping

- 0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

- 00:  $t_{DTS} = t_{CK\_INT}$
- 01:  $t_{DTS} = 2 \times t_{CK\_INT}$
- 10:  $t_{DTS} = 4 \times t_{CK\_INT}$
- 11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIMx\_ARR register is not buffered
- 1: TIMx\_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx\_CCMRx register) are set both when the counter is counting up or down.

*Note:* It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

- 0: Counter used as upcounter
- 1: Counter used as downcounter

*Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*

Bit 3 **OPM**: One-pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generate an update interrupt or DMA request if enabled.
- These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

## 16.4.2 TIM3 control register 2 (TIM3\_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.							

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

Bit 7 **TI1S**: TI1 selection

- 0: The TIMx\_CH1 pin is connected to TI1 input
  - 1: The TIMx\_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
- See also [Section 15.3.25: Interfacing with Hall sensors on page 349](#)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits permit to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT\_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred.  
(TRGO)

100: **Compare** - OC1REFC signal is used as trigger output (TRGO)

101: **Compare** - OC2REFC signal is used as trigger output (TRGO)

110: **Compare** - OC3REFC signal is used as trigger output (TRGO)

111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 3 **CCDS**: Capture/compare DMA selection

- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

### 16.4.3 TIM3 slave mode control register (TIM3\_SMCR)

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.	TS[4:3]	Res.	Res.	Res.	SMS[3]	
										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or  $\overline{ETR}$  is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).*

*It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).*

*If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK\_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

**Bits 11:8 ETF[3:0]: External trigger filter**

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{DTS}$

0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2

0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4

0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8

0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6

0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8

0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6

0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8

1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6

1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8

1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5

1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6

1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8

1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5

1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6

1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

**Bit 7 MSM: Master/Slave mode**

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)

00001: Internal Trigger 1 (ITR1)

00010: Internal Trigger 2 (ITR2)

00011: Internal Trigger 3 (ITR3)

00100: TI1 Edge Detector (TI1F\_ED)

00101: Filtered Timer Input 1 (TI1FP1)

00110: Filtered Timer Input 2 (TI2FP2)

00111: External Trigger input (ETRF)

01000: Internal Trigger 4 (ITR4)

01001: Internal Trigger 5 (ITR5)

01010: Internal Trigger 6 (ITR6)

01011: Internal Trigger 7 (ITR7)

01100: Internal Trigger 8 (ITR8)

Others: Reserved

See [Table 69: TIM3 internal trigger connection on page 447](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

0: OCREF\_CLR\_INT is unconnected.

1: OCREF\_CLR\_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

0000: Slave mode disabled - if CEN = '1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

*Note: The gated mode must not be used if TI1F\_ED is selected as the trigger input (TS=00100). Indeed, TI1F\_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.*

*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

Table 69. TIM3 internal trigger connection

Slave TIM	ITR0	ITR1	ITR2	ITR3
TIM3	TIM1	-	-	TIM14_OC1

#### 16.4.4 TIM3 DMA/Interrupt enable register (TIM3\_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

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled.

1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable  
 0: CC4 DMA request disabled.  
 1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable  
 0: CC3 DMA request disabled.  
 1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable  
 0: CC2 DMA request disabled.  
 1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable  
 0: CC1 DMA request disabled.  
 1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable  
 0: Update DMA request disabled.  
 1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable  
 0: Trigger interrupt disabled.  
 1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable  
 0: CC4 interrupt disabled.  
 1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable  
 0: CC3 interrupt disabled.  
 1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable  
 0: CC2 interrupt disabled.  
 1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable  
 0: CC1 interrupt disabled.  
 1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable  
 0: Update interrupt disabled.  
 1: Update interrupt enabled.

## 16.4.5 TIM3 status register (TIM3\_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	rc_w0
-------	--	--	-------	-------	-------	-------	--	--	-------	--	-------	-------	-------	-------	-------

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag  
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag  
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag  
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
0: No overcapture has been detected.  
1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag  
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
0: No trigger event occurred.  
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag  
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag  
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx\_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the content of TIMx\_CCR1 is greater than the content of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx\_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx\_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated:  
At overflow or underflow and if UDIS=0 in the TIMx\_CR1 register.

When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

#### 16.4.6 TIM3 event generation register (TIM3\_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								

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 TIM3 capture/compare mode register 1 (TIM3\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
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 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx\_CCER).*

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{DTS}$

0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2

0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4

0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8

0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6

0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8

0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6

0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8

1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6

1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8

1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5

1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6

1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8

1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5

1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6

1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).*

### 16.4.8 TIM3 capture/compare mode register 1 [alternate] (TIM3\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CC<sub>x</sub>S bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw
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]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx\_CCER).*

Bit 7 **OC1CE**: Output compare 1 clear enable

0: OC1Ref is not affected by the ETRF input

1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF='0) as long as TIMx\_CNT>TIMx\_CCR1 else active (OC1REF=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active. In downcounting, channel 1 is active as long as TIMx\_CNT>TIMx\_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

*Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

*Note: The OC1M[3] bit is not contiguous, located in bit 16.*

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx\_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).*

**16.4.9 TIM3 capture/compare mode register 2 (TIM3\_CCMR2)**

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
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 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx\_CCER).*

Bits 7:4 **IC3F[3:0]**: Input capture 3 filterBits 3:2 **IC3PSC[1:0]**: Input capture 3 prescalerBits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

**16.4.10 TIM3 capture/compare mode register 2 [alternate] (TIM3\_CCMR2)**

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Output compare mode:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw
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]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode

Refer to OC1M description (bits 6:4 in TIMx\_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, IC4 is mapped on TI4

10: CC4 channel is configured as input, IC4 is mapped on TI3

11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx\_CCER).*

Bit 7 **OC3CE**: Output compare 3 clear enableBits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

Refer to OC1M description (bits 6:4 in TIMx\_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enableBit 2 **OC3FE**: Output compare 3 fast enableBits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI4

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

### 16.4.11 TIM3 capture/compare enable register (TIM3\_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.*  
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)  
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)  
**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.  
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).  
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).  
CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.  
CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.
- Bit 0 **CC1E**: *Capture/Compare 1 output enable.*  
0: Capture mode disabled / OC1 is not active  
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**Table 70. Output control bit for standard OCx channels**

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

**Note:** The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO control and alternate function registers.

### 16.4.12 TIM3 counter (TIM3\_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx\_CR1 register:

- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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.
CNT[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 **CNT[15:0]**: counter value

#### 16.4.13 TIM3 counter [alternate] (TIM3\_CNT)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx\_CR1 register:

- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: counter value

#### 16.4.14 TIM3 prescaler (TIM3\_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”).

#### 16.4.15 TIM3 auto-reload register (TIM3\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

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
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

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 398](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

#### 16.4.16 TIM3 capture/compare register 1 (TIM3\_CCR1)

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.	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
CCR1[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 **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). The TIMx\_CCR1 register is read-only and cannot be programmed.

#### 16.4.17 TIM3 capture/compare register 2 (TIM3\_CCR2)

Address offset: 0x38

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
CCR2[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 **CCR2[15:0]**: Capture/Compare 2 value

**If channel CC2 is configured as output:**

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and 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). The TIMx\_CCR2 register is read-only and cannot be programmed.

#### 16.4.18 TIM3 capture/compare register 3 (TIM3\_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

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

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

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC3 output.

If channel CC3is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx CCR3 register is read-only and cannot be programmed.

#### 16.4.19 TIM3 capture/compare register 4 (TIM3\_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):  
CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.  
The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC4 output.
  2. if CC4 channel is configured as input (CC4S bits in TIMx\_CCMR4 register):  
CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx\_CCR4 register is read-only and cannot be programmed.

### 16.4.20 TIM3 DMA control register (TIM3\_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	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.

### 16.4.21 TIM3 DMA address for full transfer (TIM3\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx\_CR1 address) + (DBA + DMA index) x 4

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 16.4.22 TIM3 alternate function option register 1 (TIM3\_AF1)

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

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.

0000: ETR legacy mode

Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

### 16.4.23 TIM3 timer input selection register (TIM3\_TISEL)

Address offset: 0x68

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.	TI3SEL[3:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]	rw	rw	rw	rw	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]	rw
				rw	rw	rw	rw							rw	rw

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

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection

These bits select the TI3[0] to TI3[15] input source.

0000: TIM3\_CH3 input

Others: Reserved

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

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection

These bits select the TI2[0] to TI2[15] input source.

0000: TIM3\_CH2 input

Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection

These bits select the TI1[0] to TI1[15] input source.

0000: TIM3\_CH1 input

Others: Reserved

### 16.4.24 TIMx register map

TIMx registers are mapped as described in the table below:

**Table 71. TIM3 register map and reset values**

Offset	Register name	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
0x00	TIMx_CR1		Res																																				
	Reset value																																						
0x04	TIMx_CR2		Res																																				
	Reset value																																						
0x08	TIMx_SMCR		Res																																				
	Reset value																																						
0x0C	TIMx_DIER		Res																																				
	Reset value																																						
0x10	TIMx_SR		Res																																				
	Reset value																																						
0x14	TIMx_EGR		Res																																				
	Reset value																																						
0x18	TIMx_CCMR1 Output Compare mode		Res																																				
	Reset value																																						
	TIMx_CCMR1 Input Capture mode		Res																																				
0x1C	TIMx_CCMR2 Output Compare mode		Res																																				
	Reset value																																						
	TIMx_CCMR2 Input Capture mode		Res																																				
0x20	TIMx_CCER		Res																																				
	Reset value																																						

Table 71. TIM3 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	TIMx_CNT	UIFCPY																															
	Reset value	0																															
0x28	TIMx_PSC	Res.																															
	Reset value																																
0x2C	TIMx_ARR	Res.																															
	Reset value																																
0x30																																	
0x34	TIMx_CCR1	Res.																															
	Reset value																																
0x38	TIMx_CCR2	Res.																															
	Reset value																																
0x3C	TIMx_CCR3	Res.																															
	Reset value																																
0x40	TIMx_CCR4	Res.																															
	Reset value																																
0x44																																	
0x48	TIMx_DCR	Res.																															
	Reset value																																
0x4C	TIMx_DMAR	Res.																															
	Reset value																																
0x60	TIM3_AF1	Res.																															
	Reset value																																
0x68	TIM3_TISEL	Res.																															
	Reset value																																

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 17 General-purpose timers (TIM14)

### 17.1 TIM14 introduction

The TIM14 general-purpose timer consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM14 timer is completely independent, and does not share any resources.

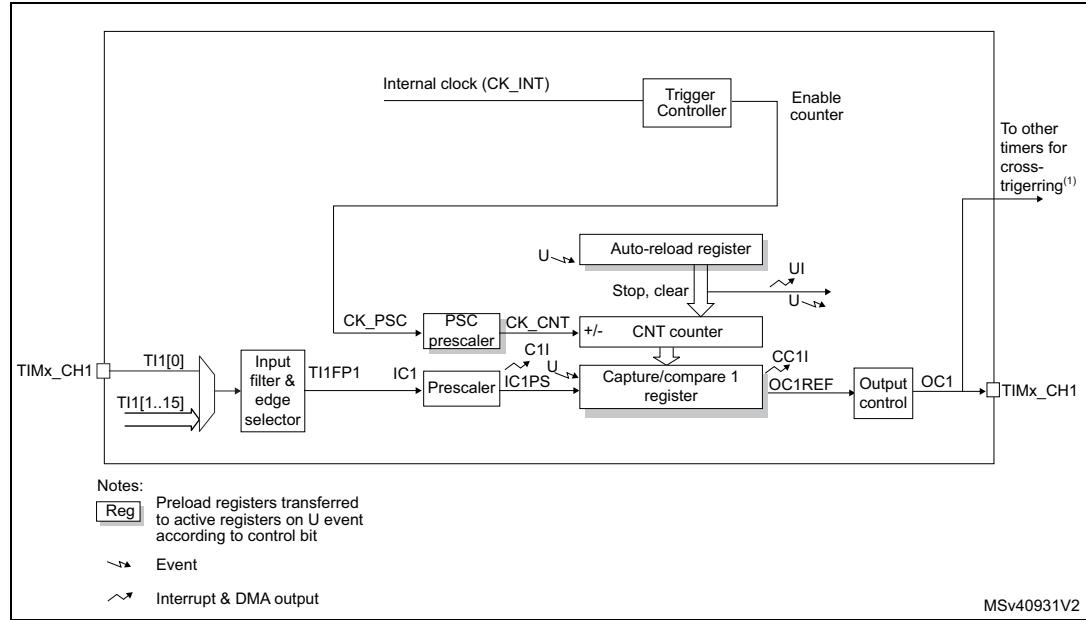
### 17.2 TIM14 main features

#### 17.2.1 TIM14 main features

The features of general-purpose timer TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

Figure 165. General-purpose timer block diagram (TIM14)



1. This signal can be used as trigger for some slave timers, see [Section 17.3.11: Using timer output as trigger for other timers \(TIM14\)](#).

MSv40931V2

## 17.3 TIM14 functional description

### 17.3.1 Time-base unit

The main block of the timer is a 16-bit up-counter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

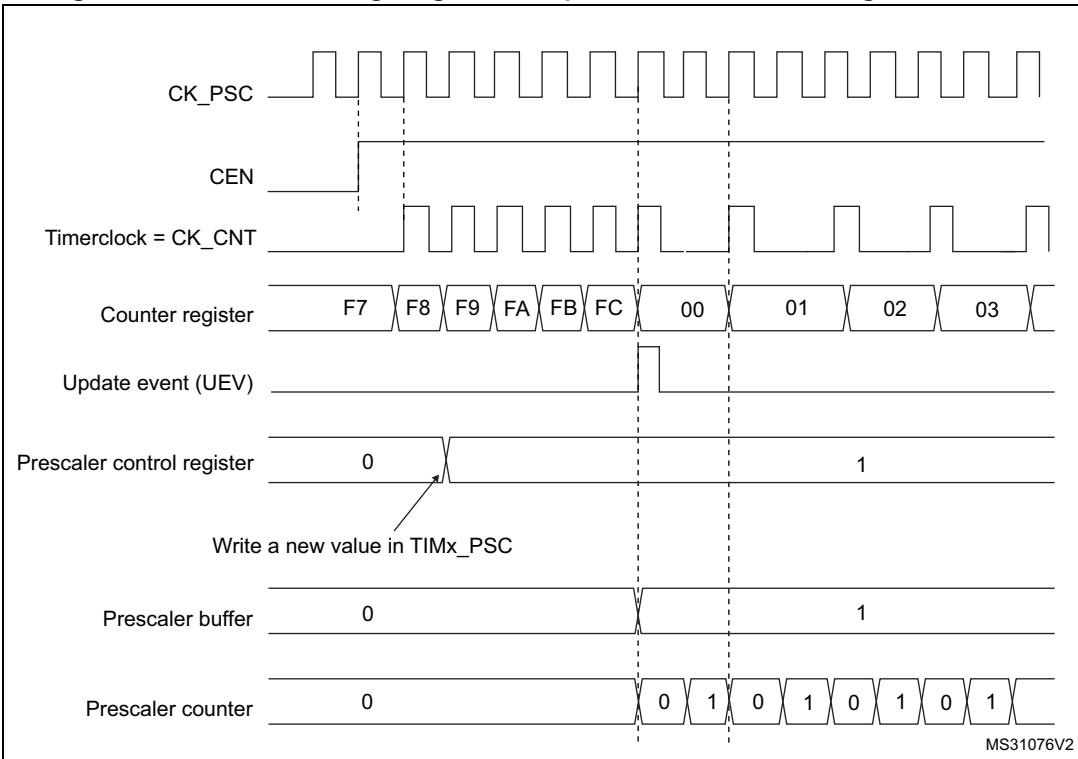
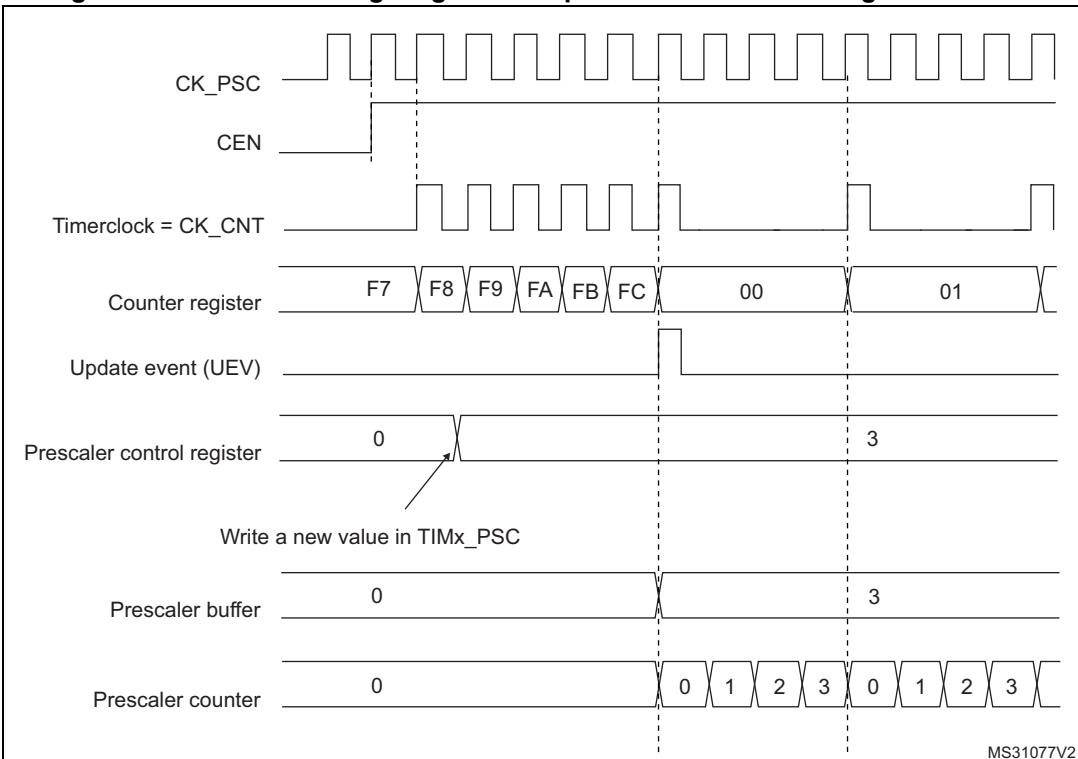
The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set.

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 166* and *Figure 167* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

**Figure 166. Counter timing diagram with prescaler division change from 1 to 2****Figure 167. Counter timing diagram with prescaler division change from 1 to 4**

### 17.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx\_EGR register (by software) also generates an update event.

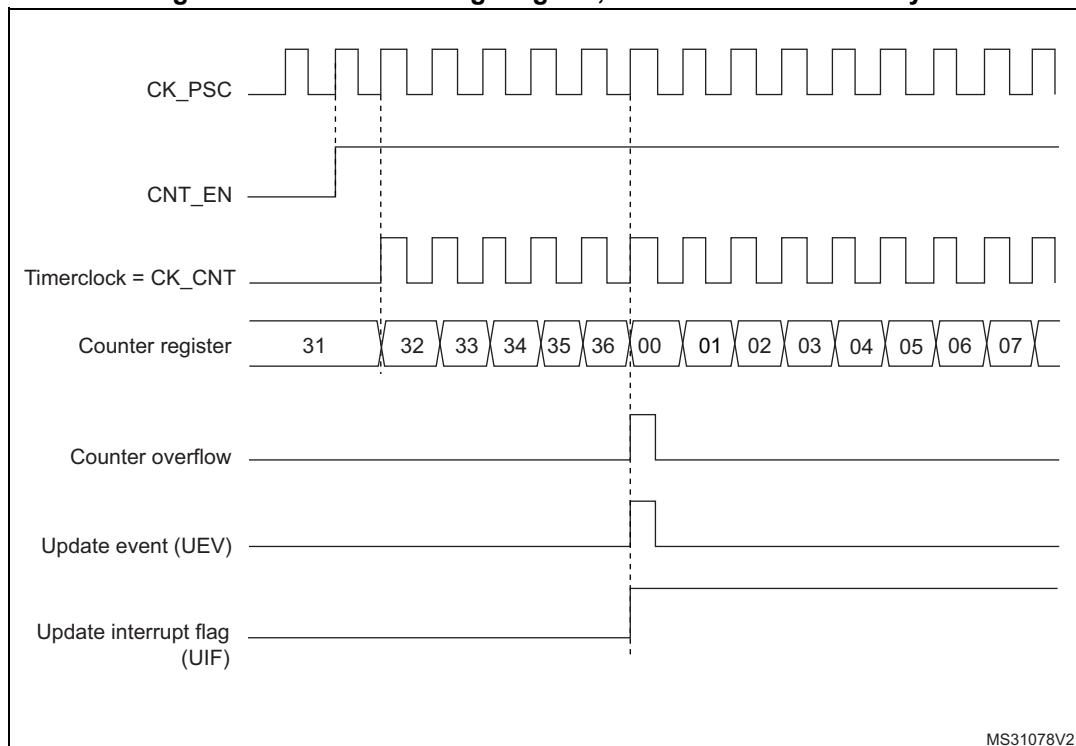
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

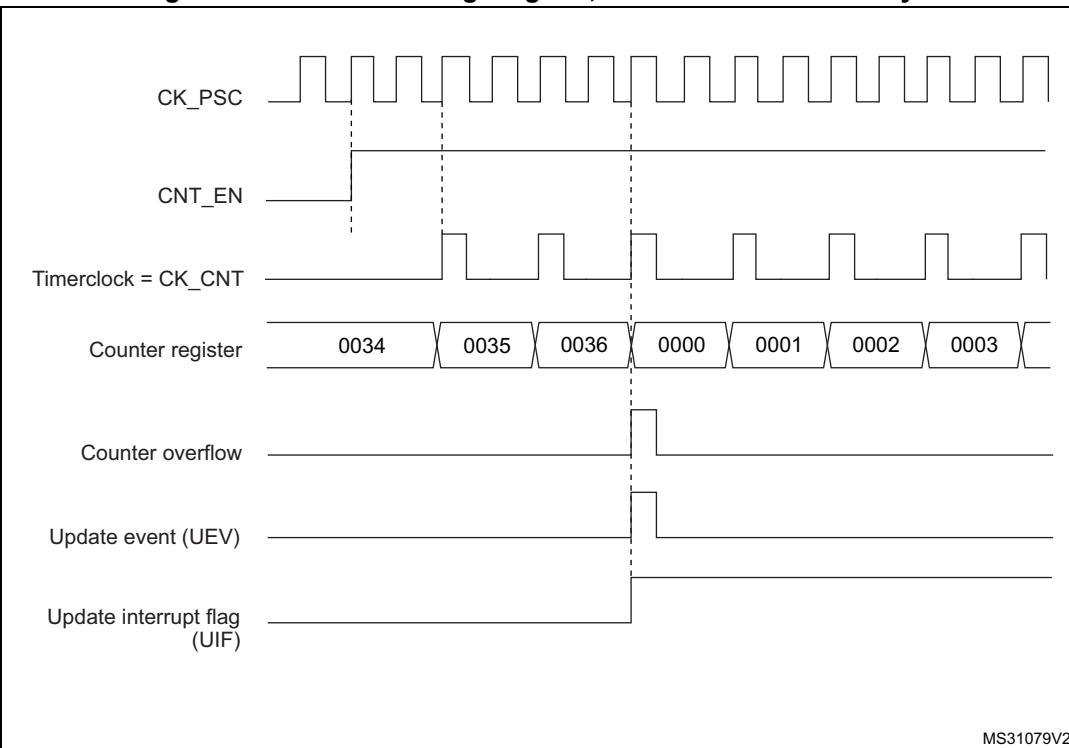
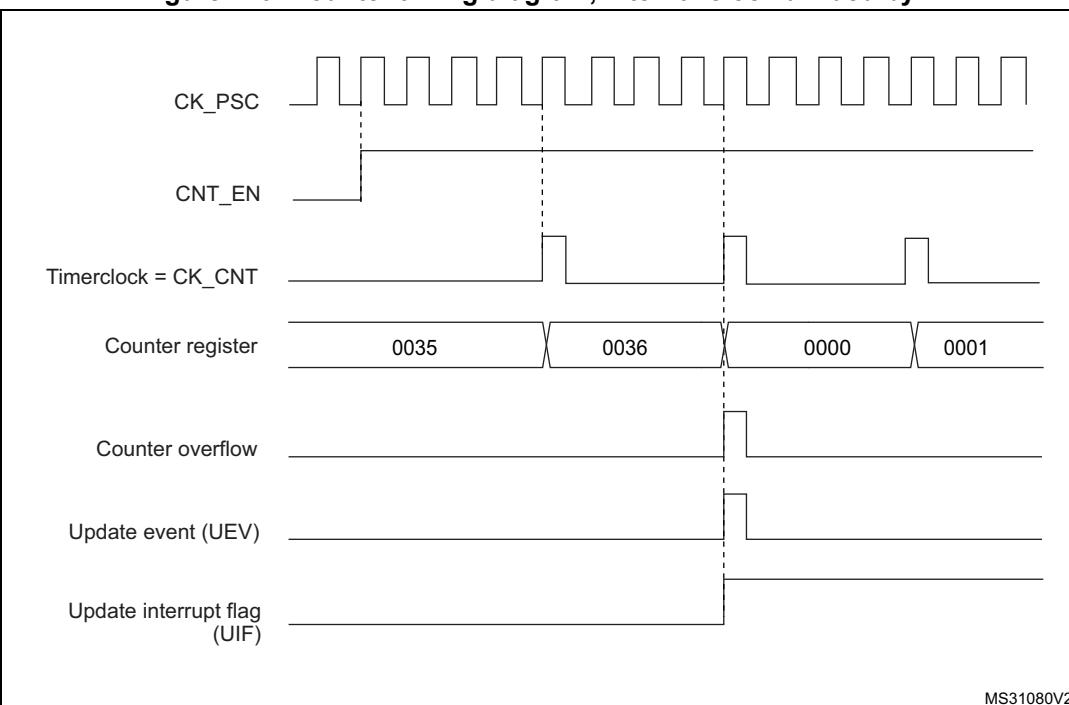
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

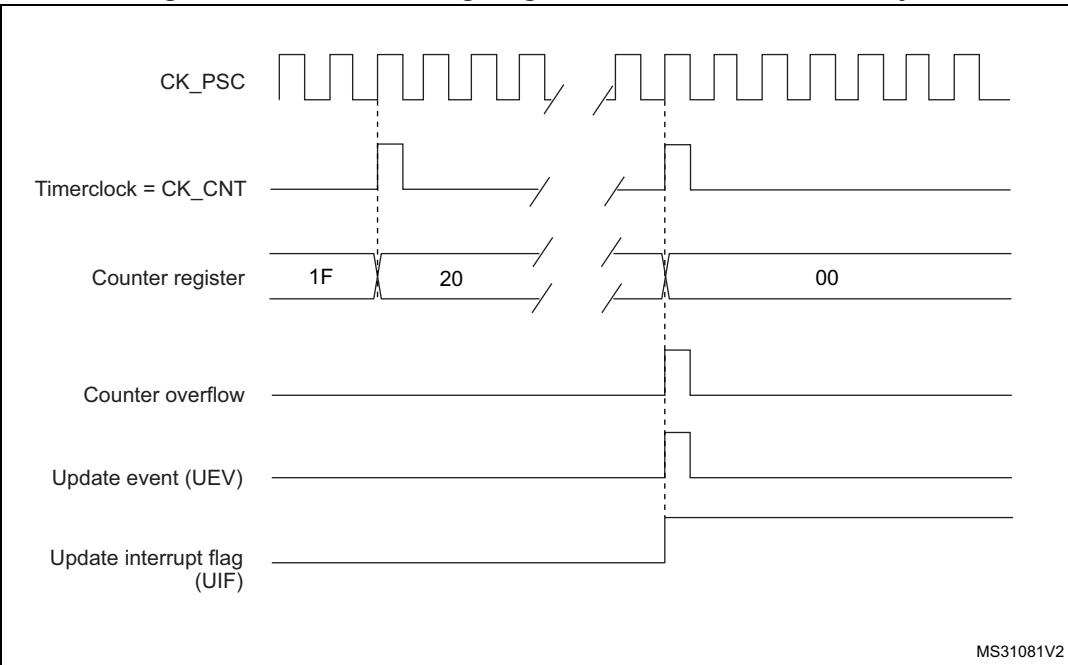
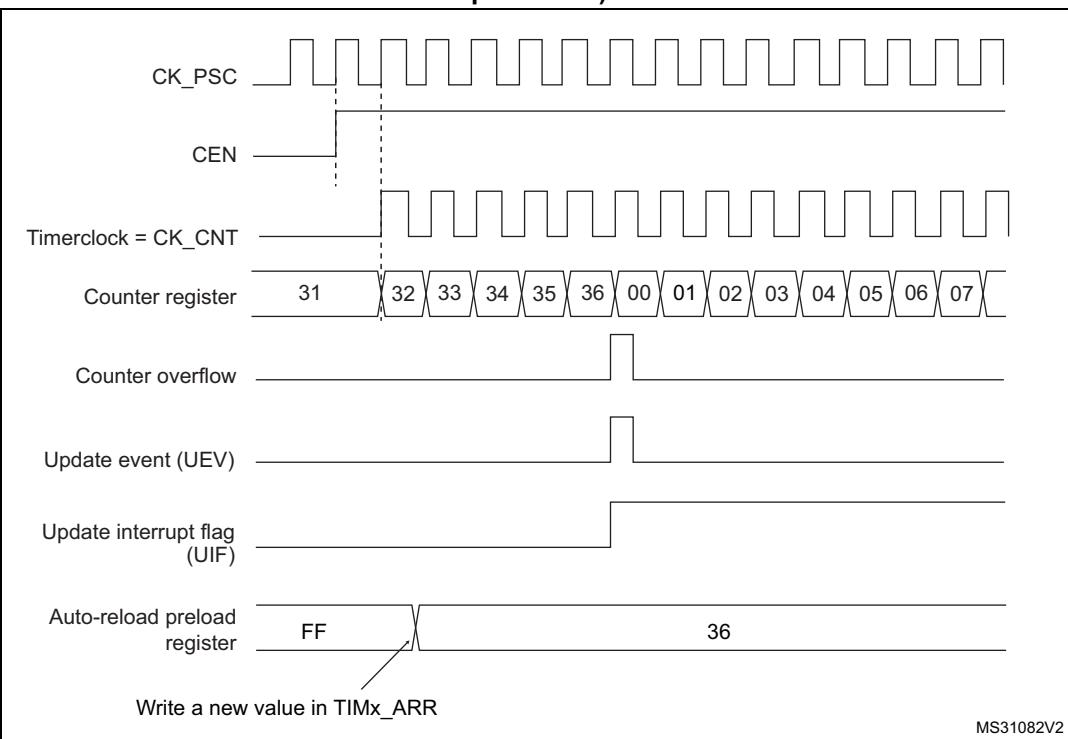
The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 168. Counter timing diagram, internal clock divided by 1**

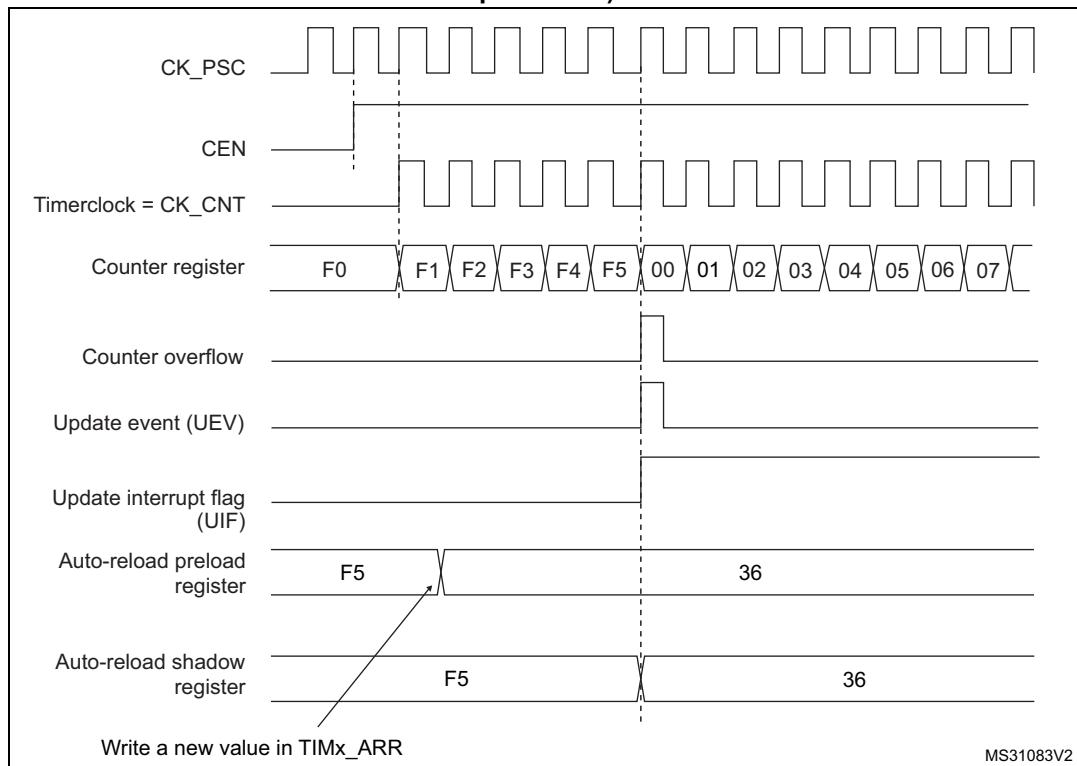


MS31078V2

**Figure 169. Counter timing diagram, internal clock divided by 2****Figure 170. Counter timing diagram, internal clock divided by 4**

**Figure 171. Counter timing diagram, internal clock divided by N****Figure 172. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**

**Figure 173. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



MS31083V2

### 17.3.3 Clock selection

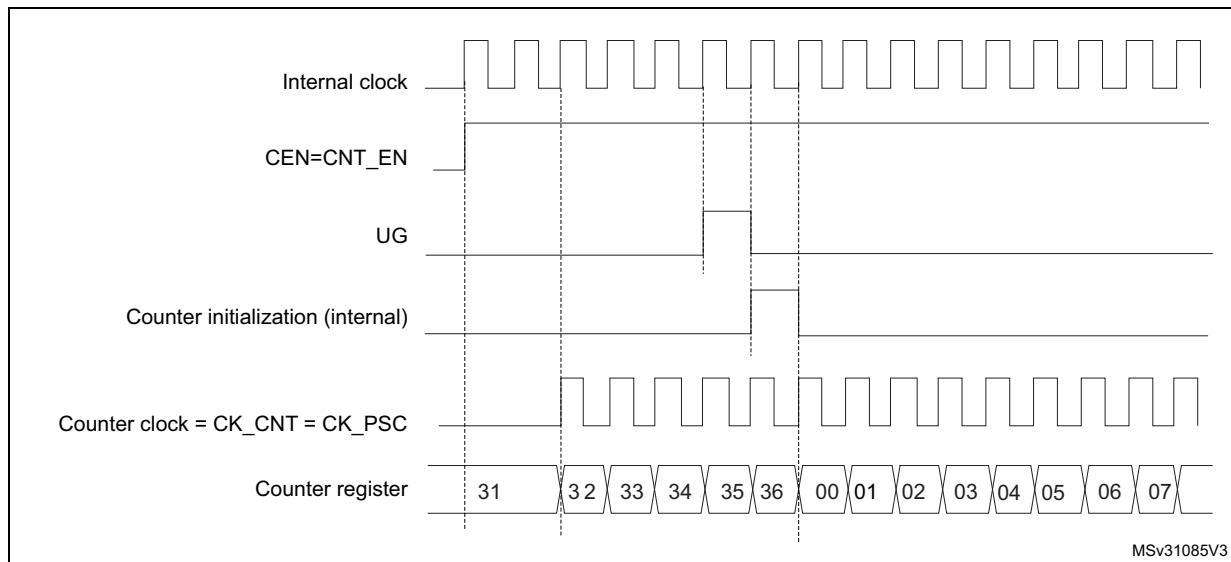
The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT)

#### Internal clock source (CK\_INT)

The internal clock source is the default clock source for TIM14.

[Figure 174](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

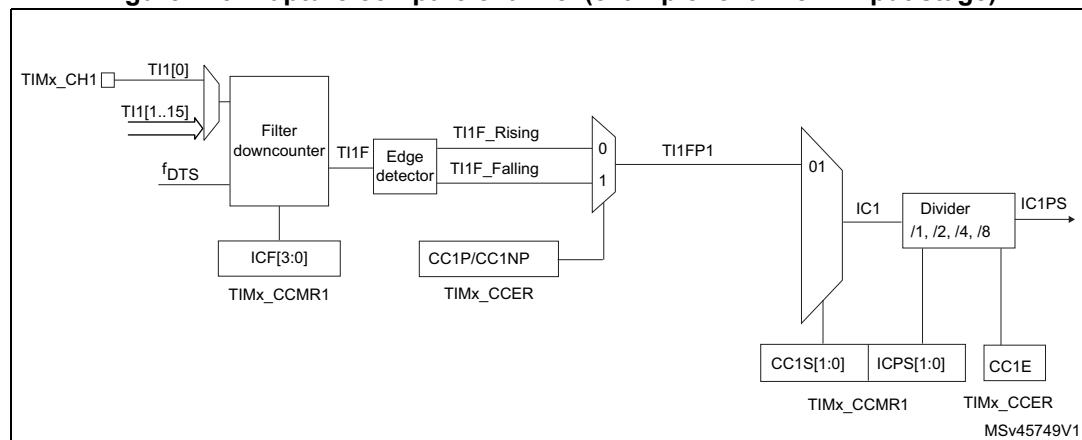
**Figure 174. Control circuit in normal mode, internal clock divided by 1**

#### 17.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

*Figure 175* to *Figure 177* 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 the capture command. It is prescaled before the capture register (ICxPS).

**Figure 175. 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 176. Capture/compare channel 1 main circuit

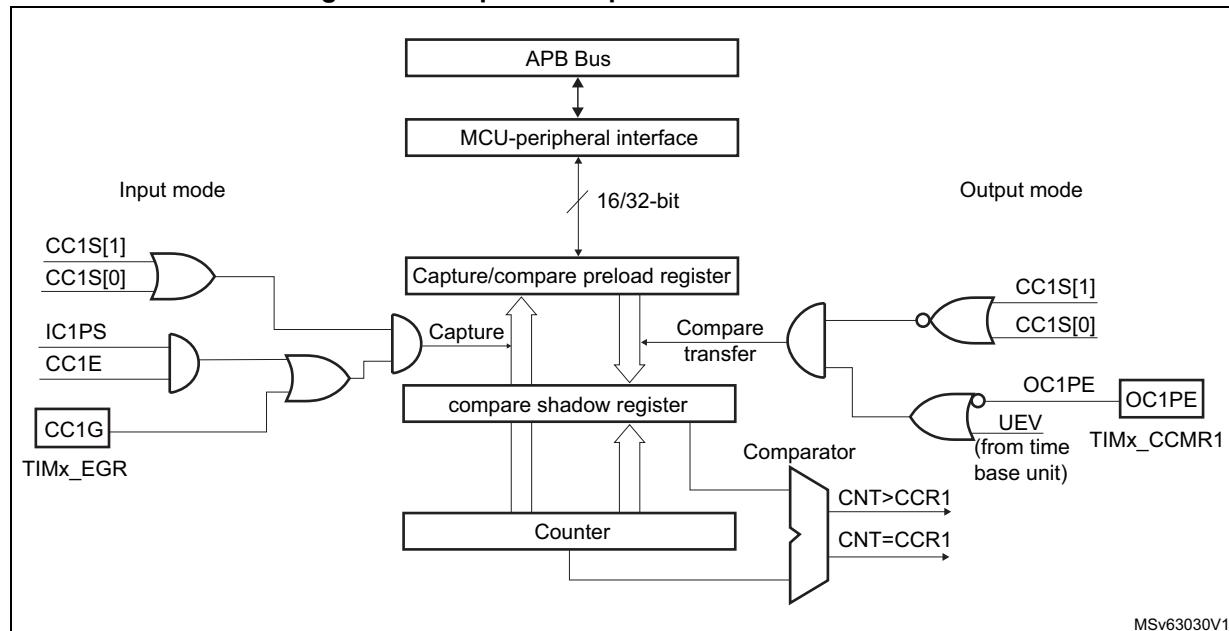
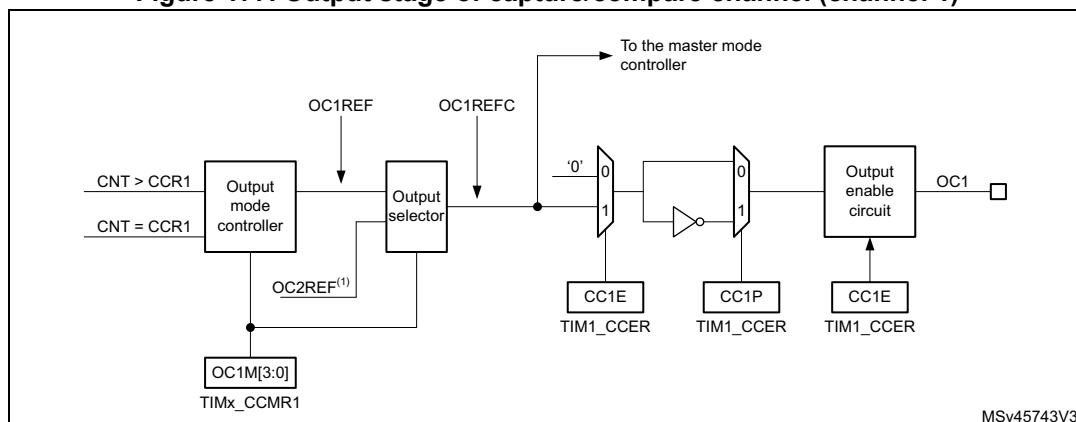


Figure 177. Output stage of capture/compare channel (channel 1)



1. Available on TIM12 only.

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.

### 17.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be

cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to '01' in the TIMx\_CCMR1 register. As soon as CC1S becomes different from '00', the channel is configured in input mode and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx\_CCMRx register if the input is one of the TIx inputs). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to '0011' in the TIMx\_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to '00' in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. 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.*

### 17.3.6 Forced output mode

In output mode (CCxS bits = '00' in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write '0101' in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP='0' (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to '0100' in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

### 17.3.7 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM='0000'), be set active (OCxM='0001'), be set inactive (OCxM='0010') or can toggle (OCxM='0011') on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx\_DIER register).

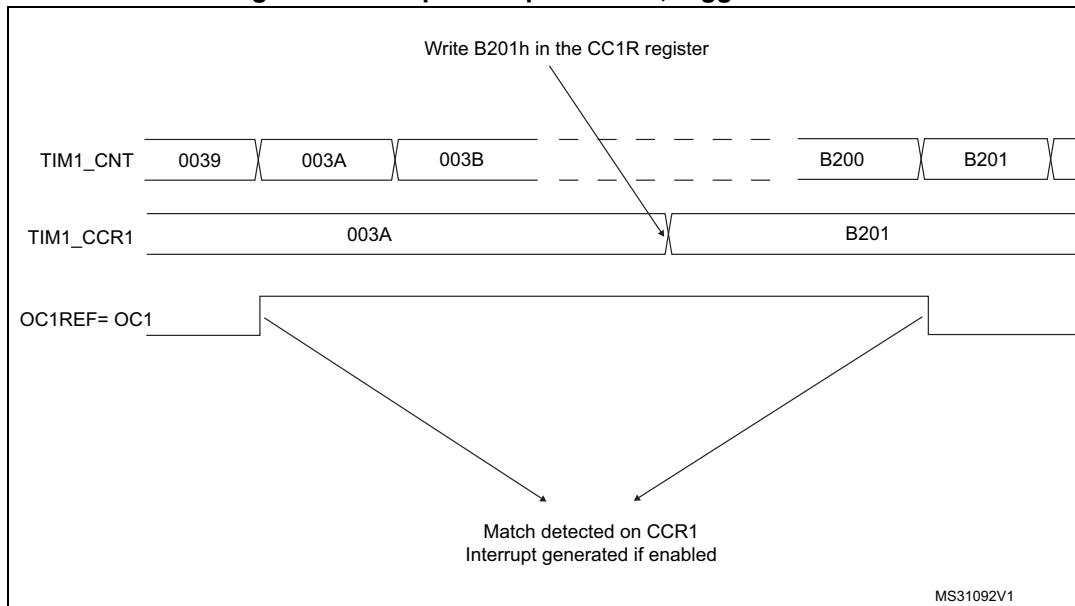
The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
  - Write OCxM = '0011' to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = '0' to disable preload register
  - Write CCxP = '0' to select active high polarity
  - Write CCxE = '1' to enable the output
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 178](#).

**Figure 178. Output compare mode, toggle on OC1.**

### 17.3.8 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the `TIMx_ARR` register and a duty cycle determined by the value of the `TIMx_CCRx` register.

The PWM mode can be selected independently on each channel (one PWM per `OCx` output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the `OCxM` bits in the `TIMx_CCMRx` register. The corresponding preload register must be enabled by setting the `OCxPE` bit in the `TIMx_CCMRx` register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the `ARPE` bit in the `TIMx_CR1` register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the `UG` bit in the `TIMx_EGR` register.

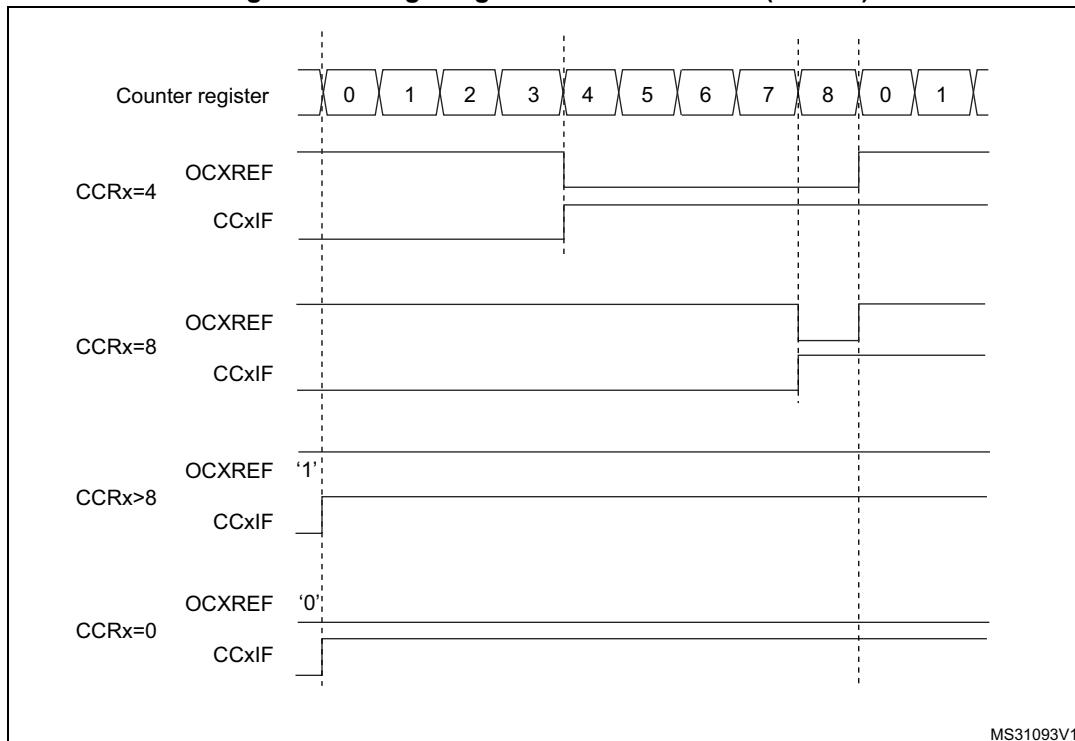
The `OCx` polarity is software programmable using the `CCxP` bit in the `TIMx_CCER` register. It can be programmed as active high or active low. The `OCx` output is enabled by the `CCxE` bit in the `TIMx_CCER` register. Refer to the `TIMx_CCERx` register description for more details.

In PWM mode (1 or 2), `TIMx_CNT` and `TIMx_CCRx` are always compared to determine whether  $\text{TIMx\_CNT} \leq \text{TIMx\_CCRx}$ .

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, we consider PWM mode 1. The reference PWM signal `OCxREF` is high as long as  $\text{TIMx\_CNT} < \text{TIMx\_CCRx}$  else it becomes low. If the compare value in `TIMx_CCRx` is greater than the auto-reload value (in `TIMx_ARR`) then `OCxREF` is held at '1'. If the compare value is 0 then `OCxRef` is held at '0'. [Figure 179](#) shows some edge-aligned PWM waveforms in an example where `TIMx_ARR`=8.

Figure 179. Edge-aligned PWM waveforms (ARR=8)



### 17.3.9 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled using the CEN bit. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

$$\text{CNT} < \text{CCRx} = \text{ARR} \text{ (in particular, } 0 < \text{CCRx)}$$

### 17.3.10 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

### 17.3.11 Using timer output as trigger for other timers (TIM14)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx\_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.

For instance, if the destination's timer CK\_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

### 17.3.12 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M0+ core halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBG module. For more details, refer to [Section 26.9.2: Debug support for timers, watchdog, and I2C](#).

## 17.4 TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

### 17.4.1 TIM14 control register 1 (TIM14\_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

- 0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and sampling clock used by the digital filters (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 6:4 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 update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

**Bit 1 UDIS:** Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

- Counter overflow
- Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

**Bit 0 CEN:** Counter enable

0: Counter disabled

1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

**17.4.2 TIM14 Interrupt enable register (TIM14\_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													

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

**17.4.3 TIM14 status register (TIM14\_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.	Res.	Res.	Res.	CC1OF	Res.	CC1IF	UIF						

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx\_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the content of TIMx\_CCR1 is greater than the content of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx\_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx\_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow 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.

#### 17.4.4 TIM14 event generation register (TIM14\_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													

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.

### 17.4.5 TIM14 capture/compare mode register 1 (TIM14\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]								
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{DTS}$
- 0001:  $f_{SAMPLING} = f_{CK\_INT}$ , N=2
- 0010:  $f_{SAMPLING} = f_{CK\_INT}$ , N=4
- 0011:  $f_{SAMPLING} = f_{CK\_INT}$ , N=8
- 0100:  $f_{SAMPLING} = f_{DTS}/2$ , N=6
- 0101:  $f_{SAMPLING} = f_{DTS}/2$ , N=8
- 0110:  $f_{SAMPLING} = f_{DTS}/4$ , N=6
- 0111:  $f_{SAMPLING} = f_{DTS}/4$ , N=8
- 1000:  $f_{SAMPLING} = f_{DTS}/8$ , N=6
- 1001:  $f_{SAMPLING} = f_{DTS}/8$ , N=8
- 1010:  $f_{SAMPLING} = f_{DTS}/16$ , N=5
- 1011:  $f_{SAMPLING} = f_{DTS}/16$ , N=6
- 1100:  $f_{SAMPLING} = f_{DTS}/16$ , N=8
- 1101:  $f_{SAMPLING} = f_{DTS}/32$ , N=5
- 1110:  $f_{SAMPLING} = f_{DTS}/32$ , N=6
- 1111:  $f_{SAMPLING} = f_{DTS}/32$ , N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).

The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: Reserved

11: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).*

**17.4.6 TIM14 capture/compare mode register 1 [alternate] (TIM14\_CCMR1)**

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]									
															rw
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]											
									rw	rw	rw	rw	rw	rw	rw

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

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

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT = TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx\_CNT < TIMx\_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx\_CNT < TIMx\_CCR1 else active

Others: Reserved

*Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.*

*Note: The OC1M[3] bit is not contiguous, located in bit 16.*

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx\_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: Reserved.

11: Reserved.

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER).*

#### 17.4.7 TIM14 capture/compare enable register (TIM14\_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											

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.

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0:This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**Table 72. Output control bit for standard OCx channels**

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

**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.

#### 17.4.8 TIM14 counter (TIM14\_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx\_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

### 17.4.9 TIM14 prescaler (TIM14\_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”).

### 17.4.10 TIM14 auto-reload register (TIM14\_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

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 17.3.1: Time-base unit on page 469](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 17.4.11 TIM14 capture/compare register 1 (TIM14\_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).

### 17.4.12 TIM14 timer input selection register (TIM14\_TISEL)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TI1SEL[3:0]														

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM14\_CH1 input

0001: RTC CLK

0010: HSE/32

0011: MCO

0100: MCO2

Others: Reserved

### 17.4.13 TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 73. TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	UIFREMA	CKD [1:0]	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN																					
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04 to 0x08	Reserved																																
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE																						
	Reset value																						0	0	0	0	0	0	0	0	0	0	

Table 73. TIM14 register map and reset values (continued)

Offset	Register name	Reset value
0x10	TIMx_SR	Res. 31
	Reset value	Res. 30
0x14	TIMx_EGR	Res. 29
	Reset value	Res. 28
0x18	TIMx_CCMR1 Output compare mode	Res. 27
	Reset value	Res. 26
0x18	TIMx_CCMR1 Input capture mode	Res. 25
	Reset value	Res. 24
0x1C	Reserved	Res.
0x20	TIMx_CCER	Res. 20
	Reset value	Res. 19
0x24	TIMx_CNT	Res. 18
	Reset value	Res. 17
0x28	TIMx_PSC	OC1M[3] Res. 16
	Reset value	0 Res. 15
0x2C	TIMx_ARR	CNT[15:0] Res. 14
	Reset value	Res. 13
0x30	Reserved	Res. 12
0x34	TIMx_CCR1	IC1F[3:0] Res. 11
	Reset value	0 0 0 0 Res. 10
0x38 to 0x64	Reserved	OC1M[2:0] Res. 9
0x68	TIM14_TISEL	IC1PSC[1:0] Res. 8
	Reset value	0 CC1NP Res. 7
		CC1S[1:0] Res. 6
		0 CC1IP Res. 5
		0 CC1E Res. 4
		0 UG Res. 3
		0 UIF Res. 2
		0 CC1IF Res. 1
		0 UG Res. 0

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 18 General-purpose timers (TIM16/TIM17)

### 18.1 TIM16/TIM17 introduction

The TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

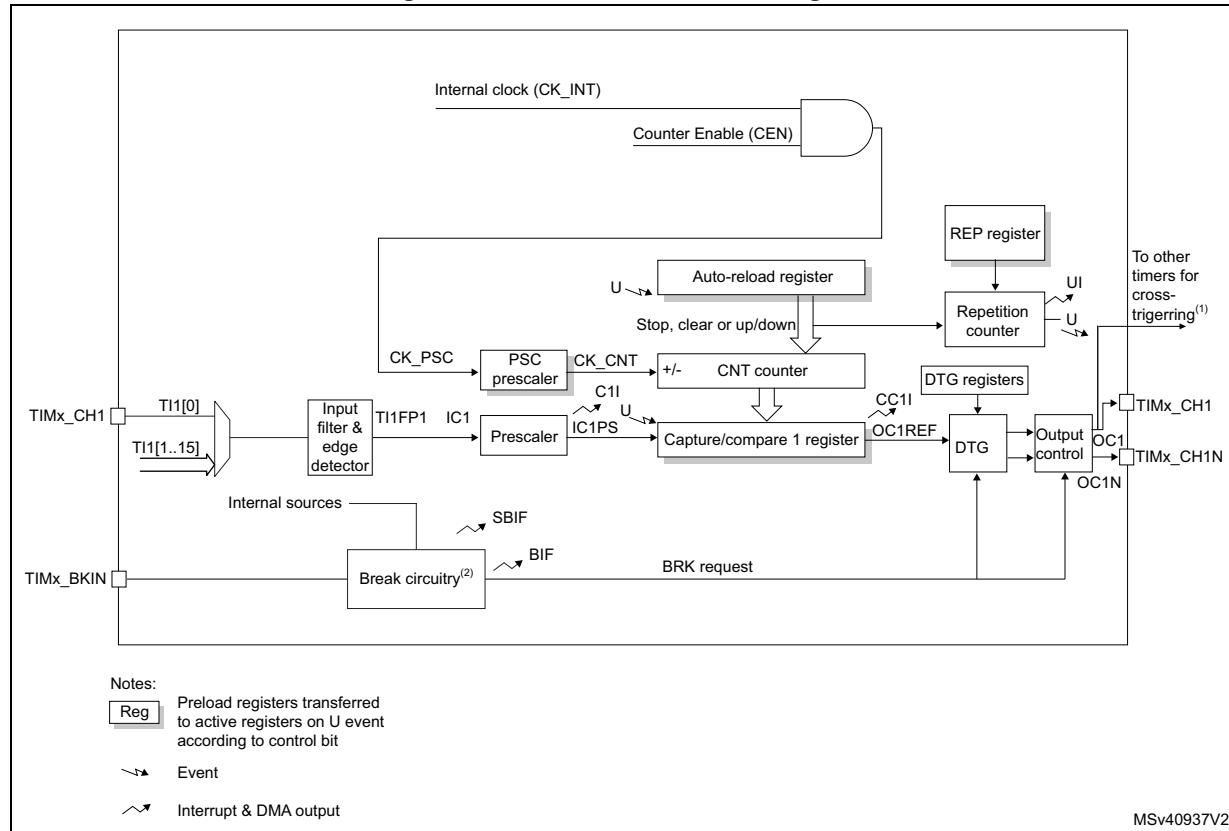
The TIM16/TIM17 timers are completely independent, and do not share any resources.

### 18.2 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 180. TIM16/TIM17 block diagram



1. This signal can be used as trigger for some slave timer, see [Section 18.3.18: Using timer output as trigger for other timers \(TIM16/TIM17\)](#).
2. The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 5.2.6: Clock security system \(CSS\)](#)
  - SRAM parity error signal
  - Cortex®-M0+ LOCKUP (Hardfault) output

## 18.3 TIM16/TIM17 functional description

### 18.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

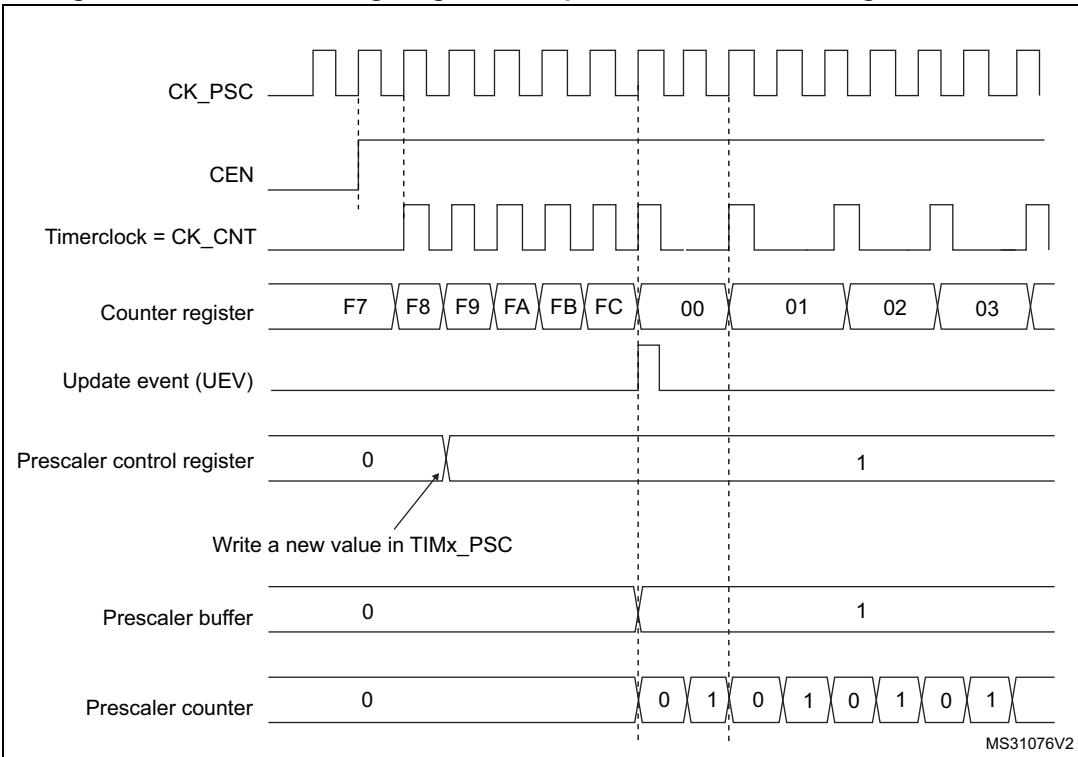
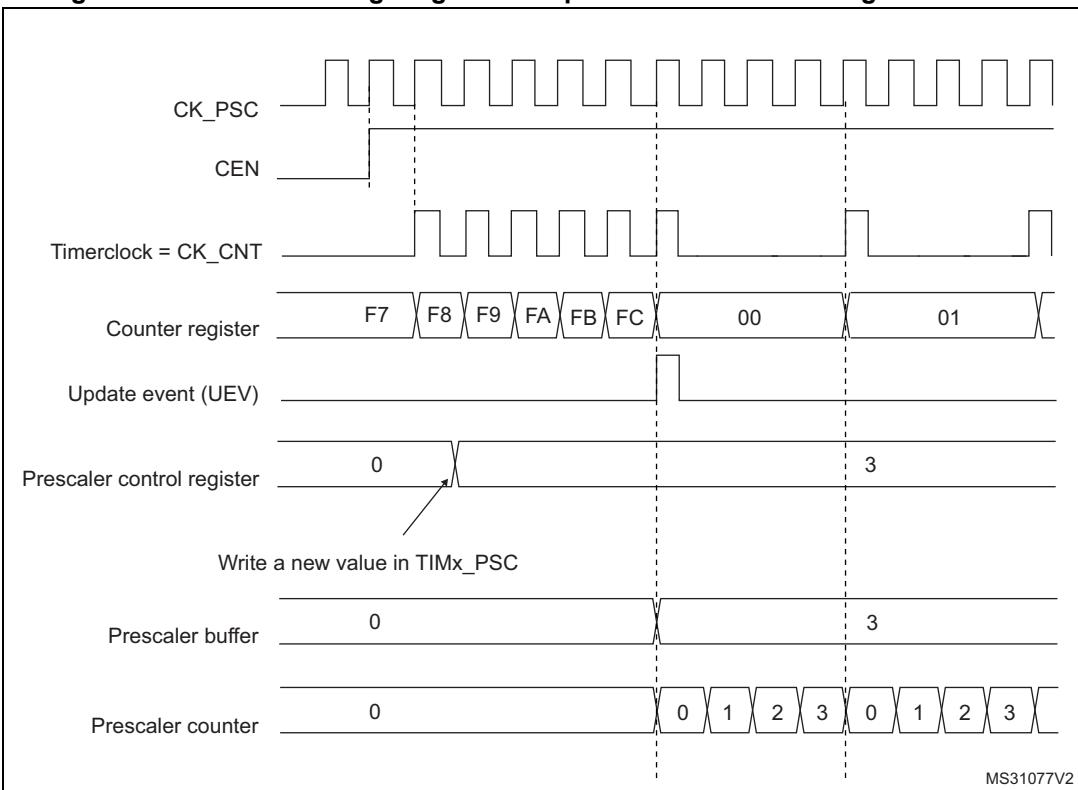
The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 181* and *Figure 182* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 181. Counter timing diagram with prescaler division change from 1 to 2****Figure 182. Counter timing diagram with prescaler division change from 1 to 4**

### 18.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR). Else the update event is generated at each counter overflow.

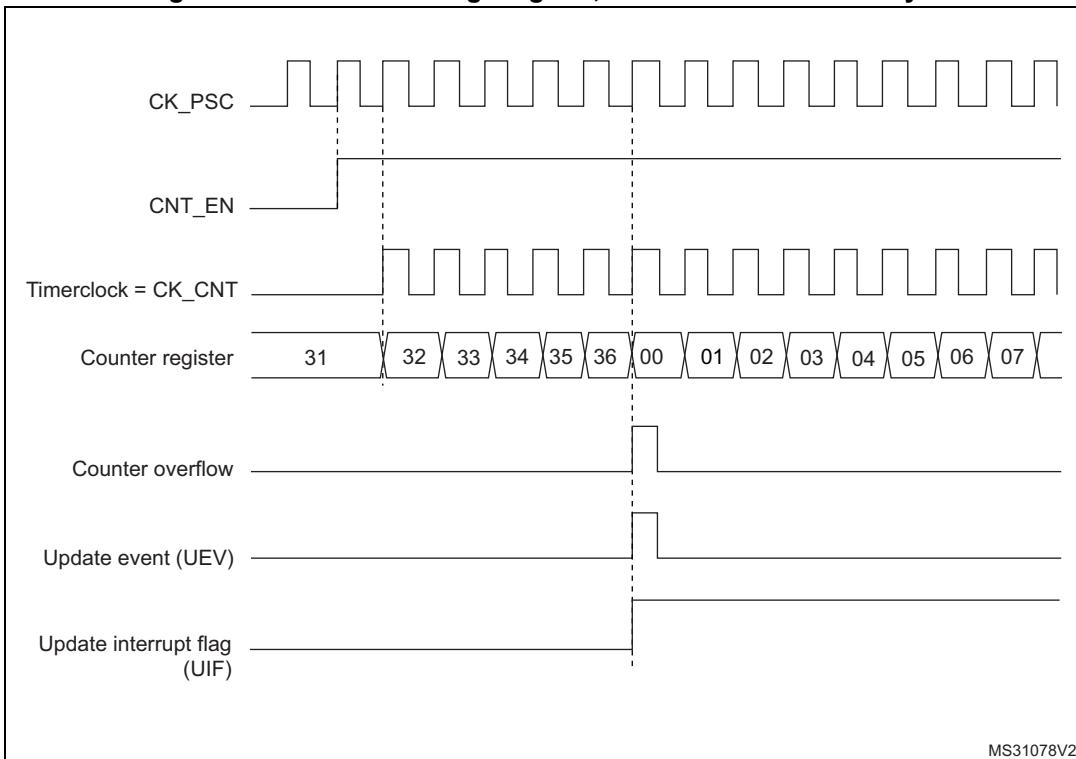
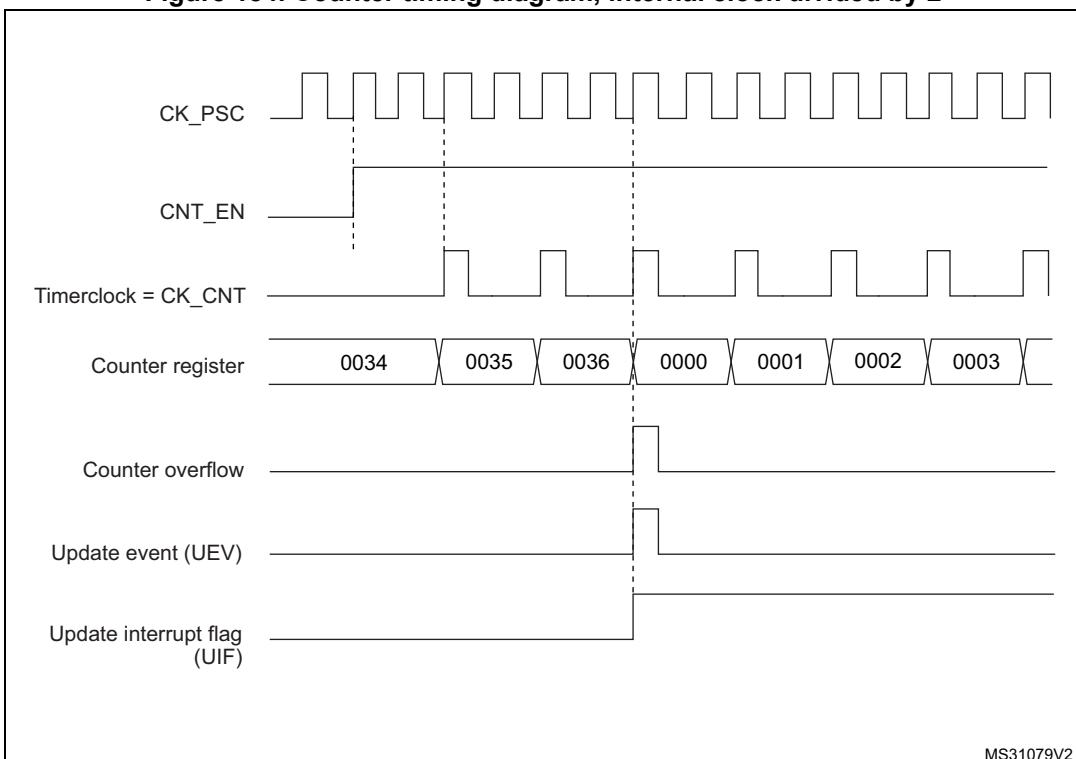
Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

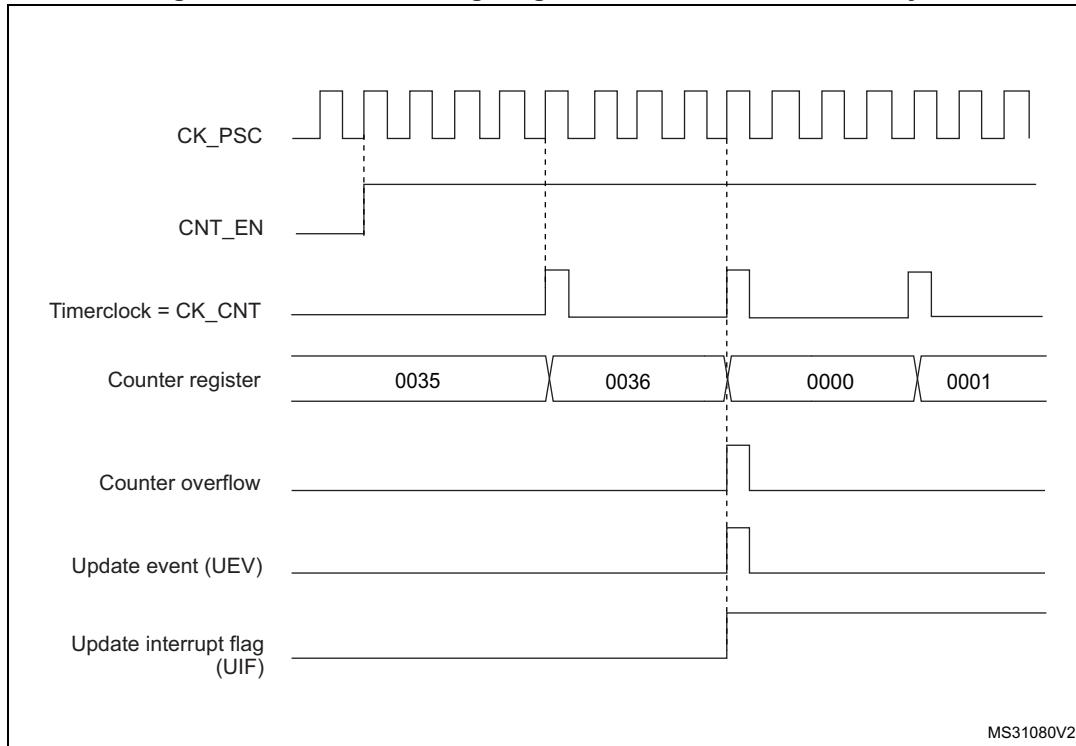
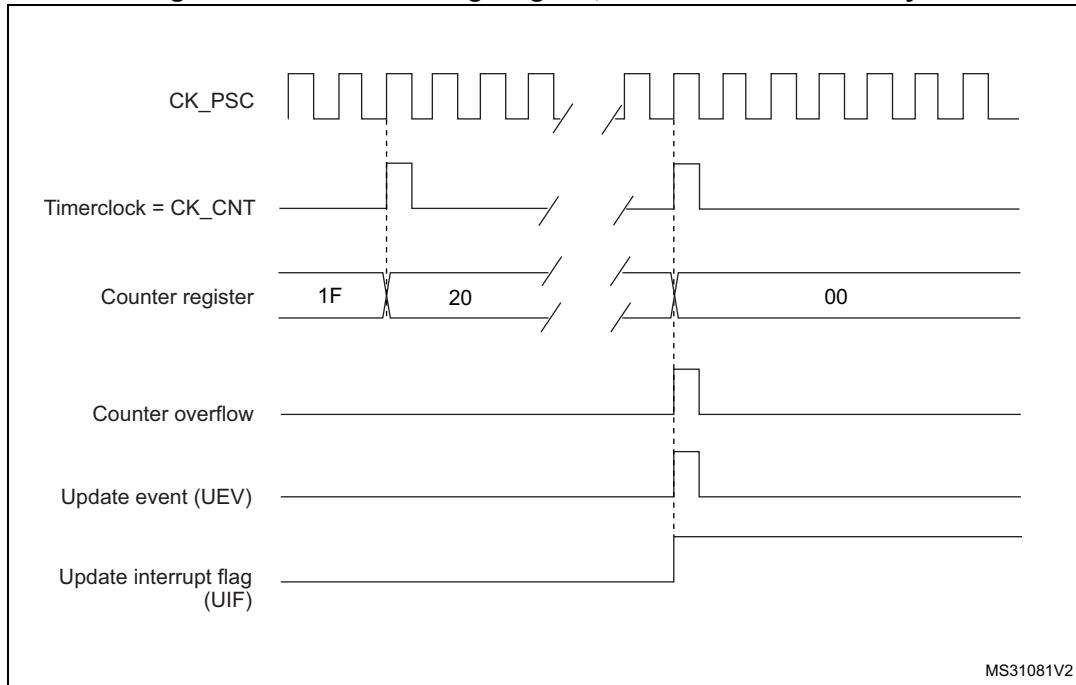
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

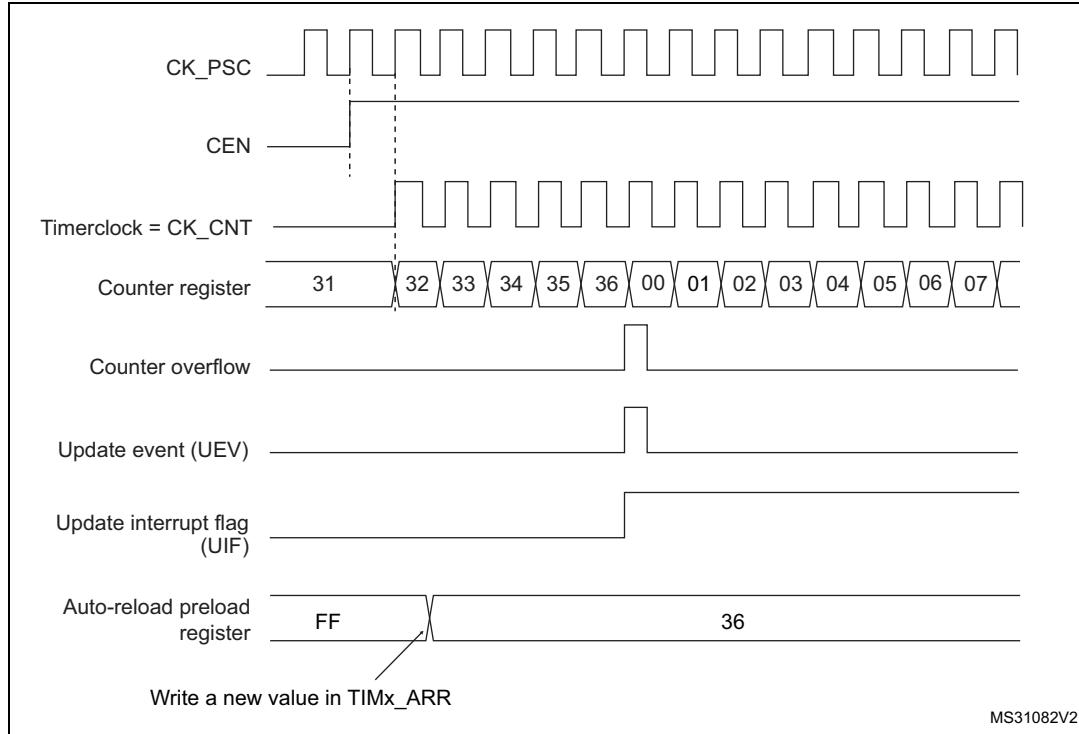
- The repetition counter is reloaded with the content of TIMx\_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

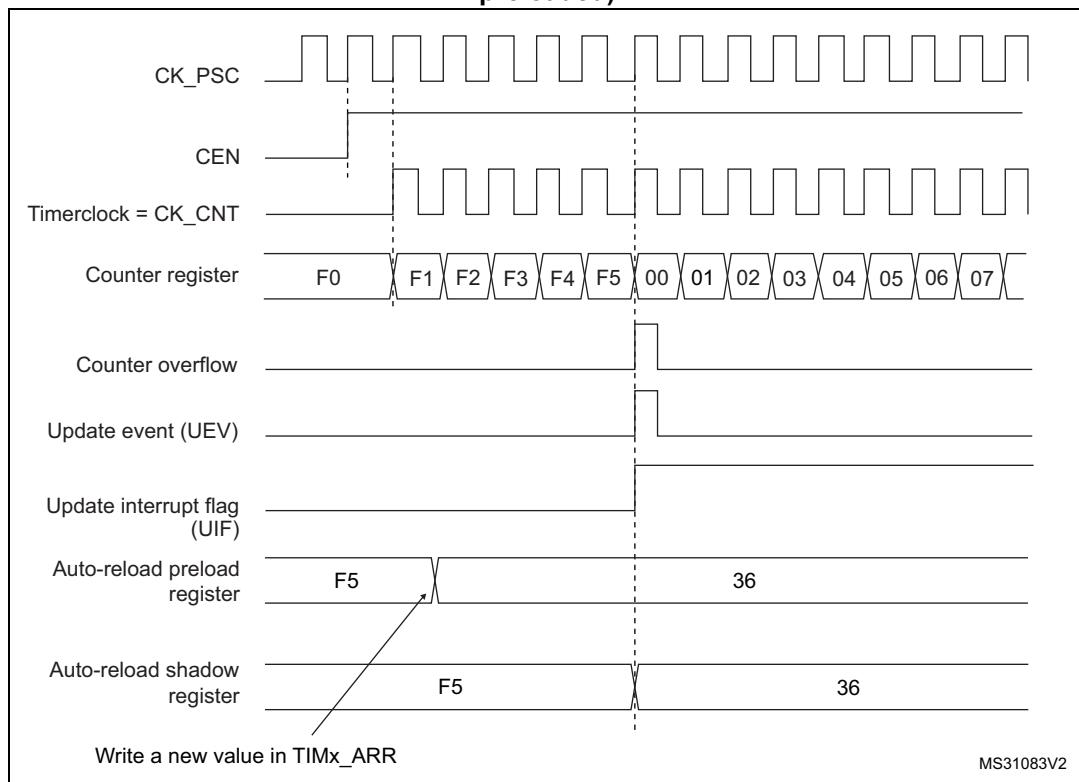
**Figure 183. Counter timing diagram, internal clock divided by 1****Figure 184. Counter timing diagram, internal clock divided by 2**

**Figure 185. Counter timing diagram, internal clock divided by 4****Figure 186. Counter timing diagram, internal clock divided by N**

**Figure 187. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 188. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 18.3.3 Repetition counter

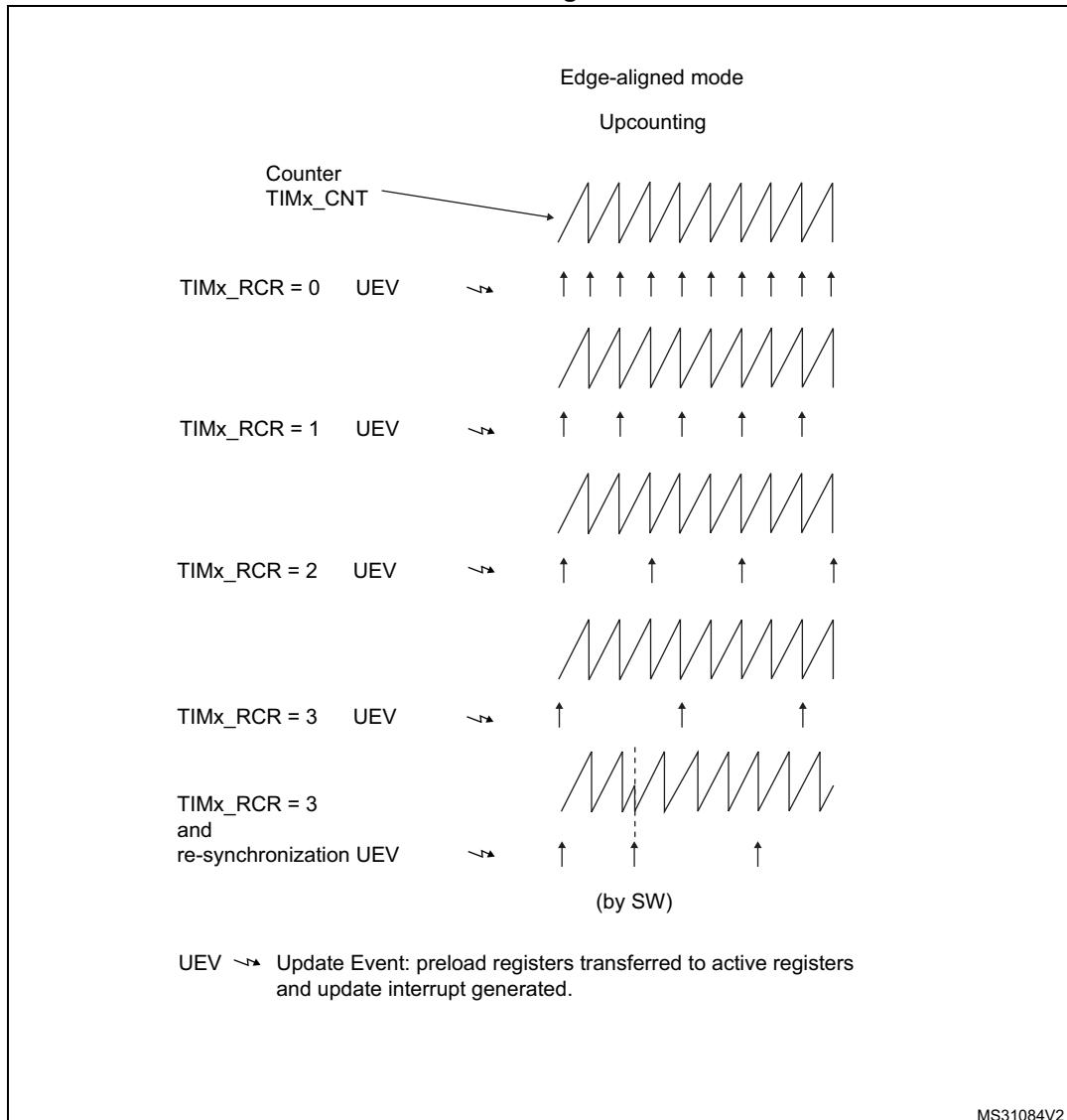
*Section 18.3.1: Time-base unit* describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC prescaler register, but also TIMx\_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to [Figure 189](#)). 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 189. Update rate examples depending on mode and TIMx\_RCR register settings**



### 18.3.4 Clock selection

The counter clock can be provided by the following clock sources:

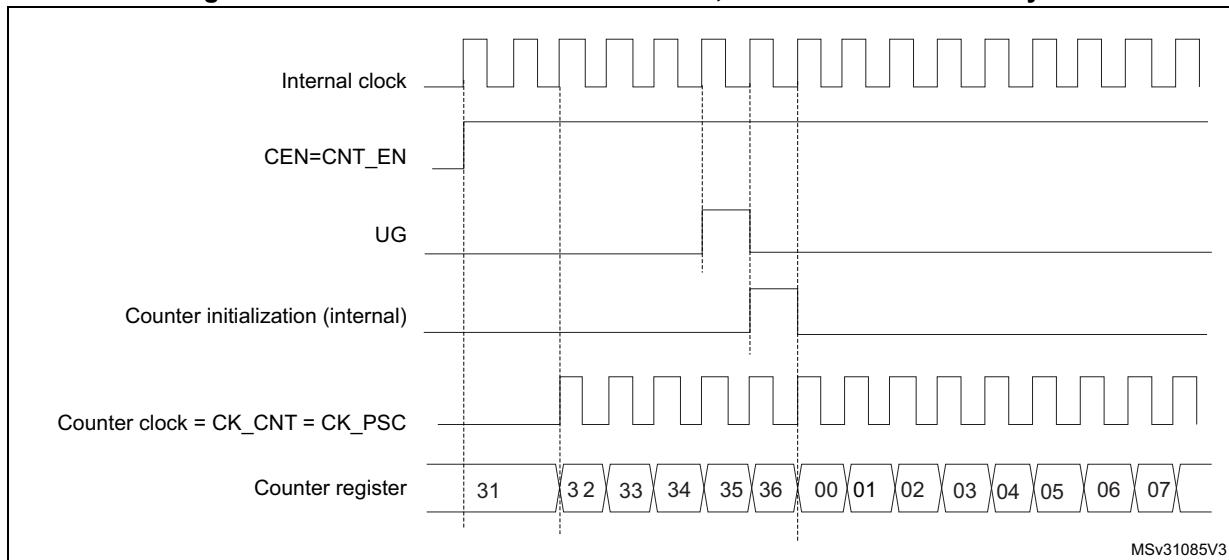
- Internal clock (CK\_INT)
  - External clock mode1: external input pin

## Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

*Figure 190* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

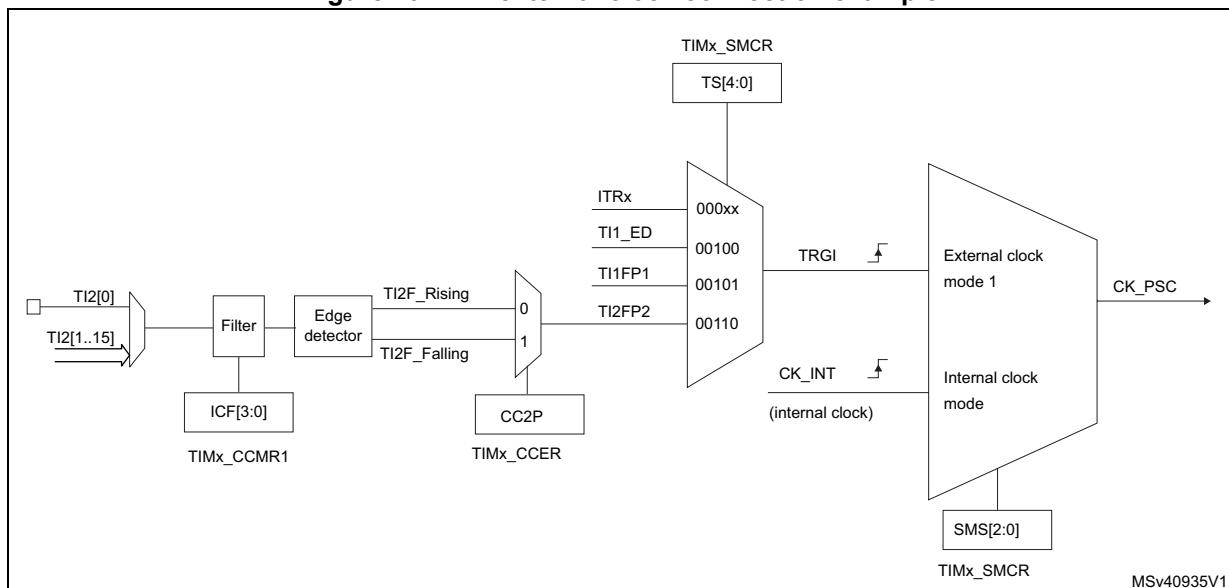
**Figure 190. Control circuit in normal mode, internal clock divided by 1**



### 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 191. 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:

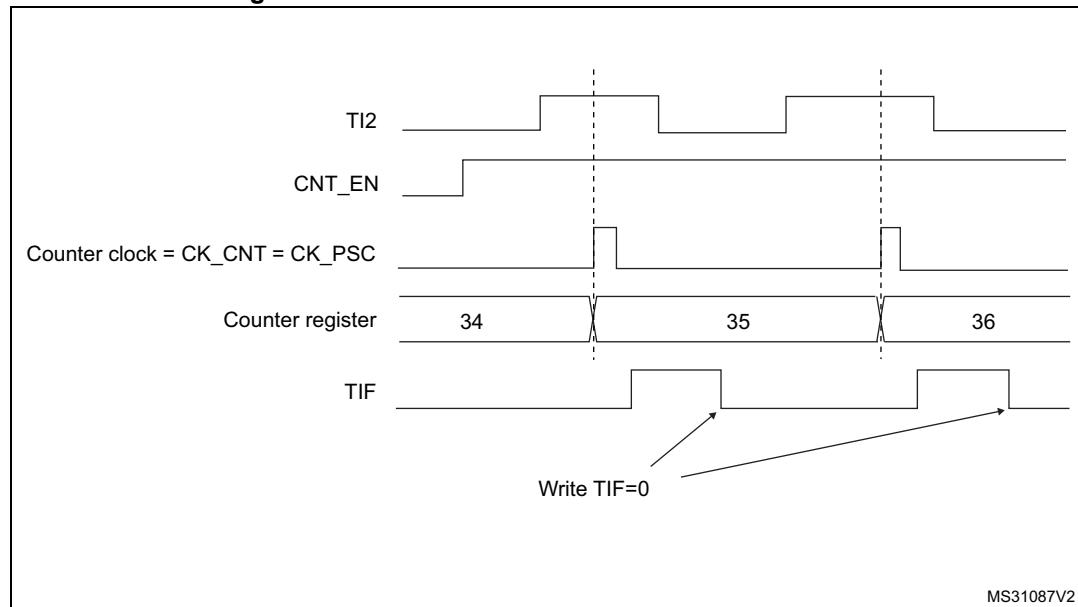
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx\_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx\_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

*Note:* The capture prescaler is not used for triggering, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 192. Control circuit in external clock mode 1

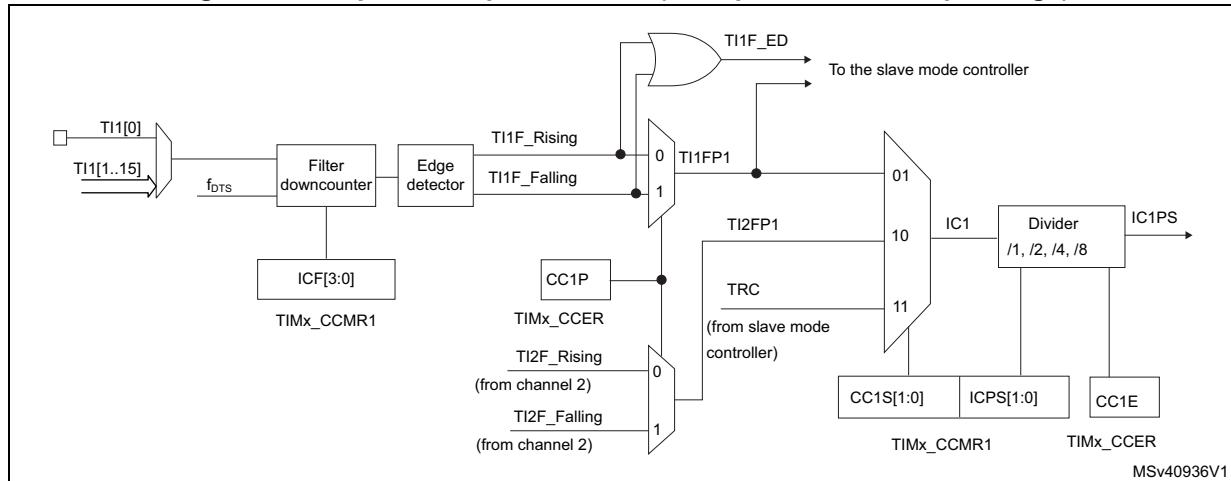


### 18.3.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

[Figure 193](#) to [Figure 195](#) 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 193. 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.

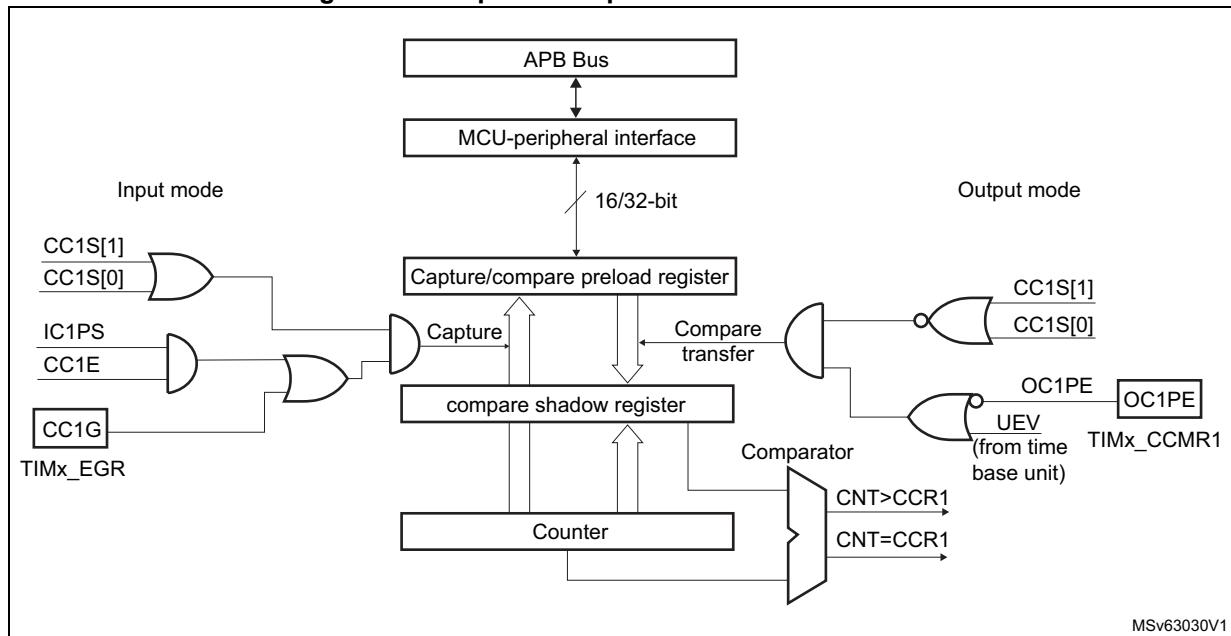
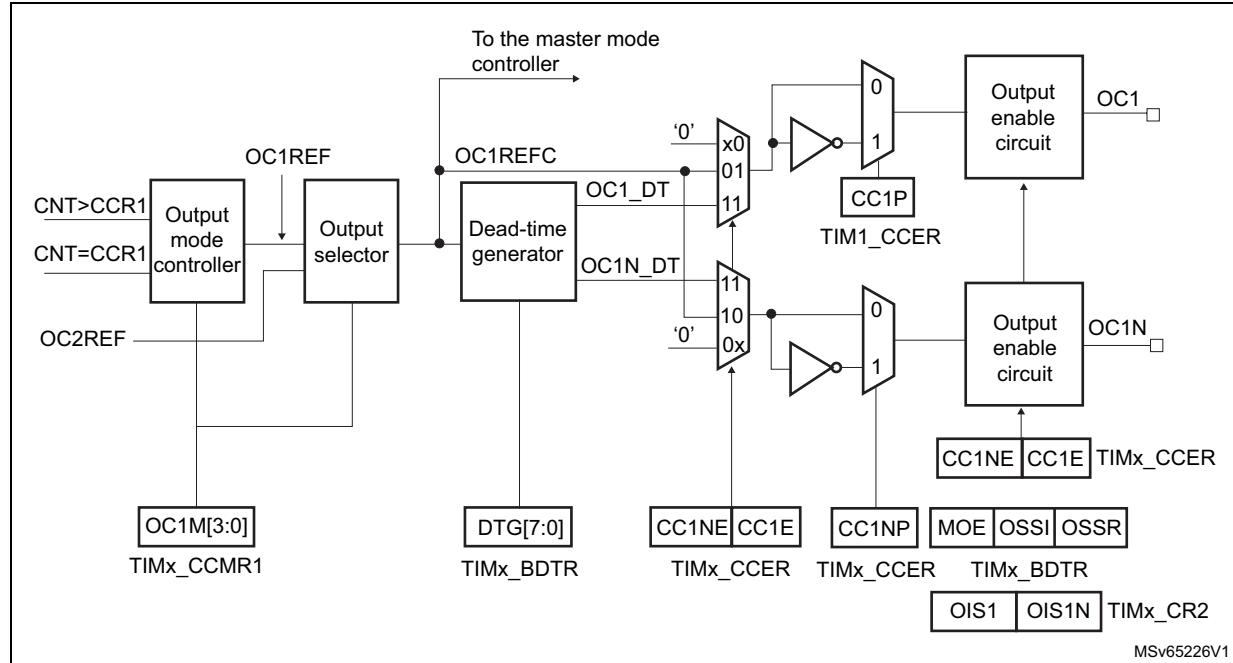
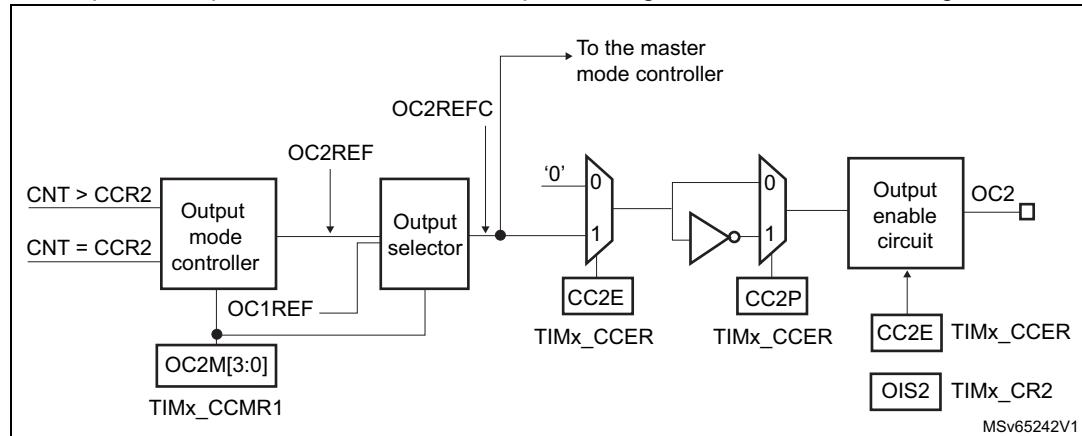
**Figure 194. Capture/compare channel 1 main circuit**

Figure 195. 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.

### 18.3.6 Input capture mode

In Input capture mode, the Capture/Compare registers (**TIMx\_CCRx**) are used to latch the value of the counter after a transition detected by the corresponding **ICx** signal. When a capture occurs, the corresponding **CCxIF** flag (**TIMx\_SR** register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the **CCxIF** flag was already high, then the over-capture flag **CCxOF** (**TIMx\_SR** register) is set. **CCxIF** can be

cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  frequency). Then write IC1F bits to 0011 in the TIMx\_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note:

*IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.*

1. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).

### 18.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 18.3.8 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.

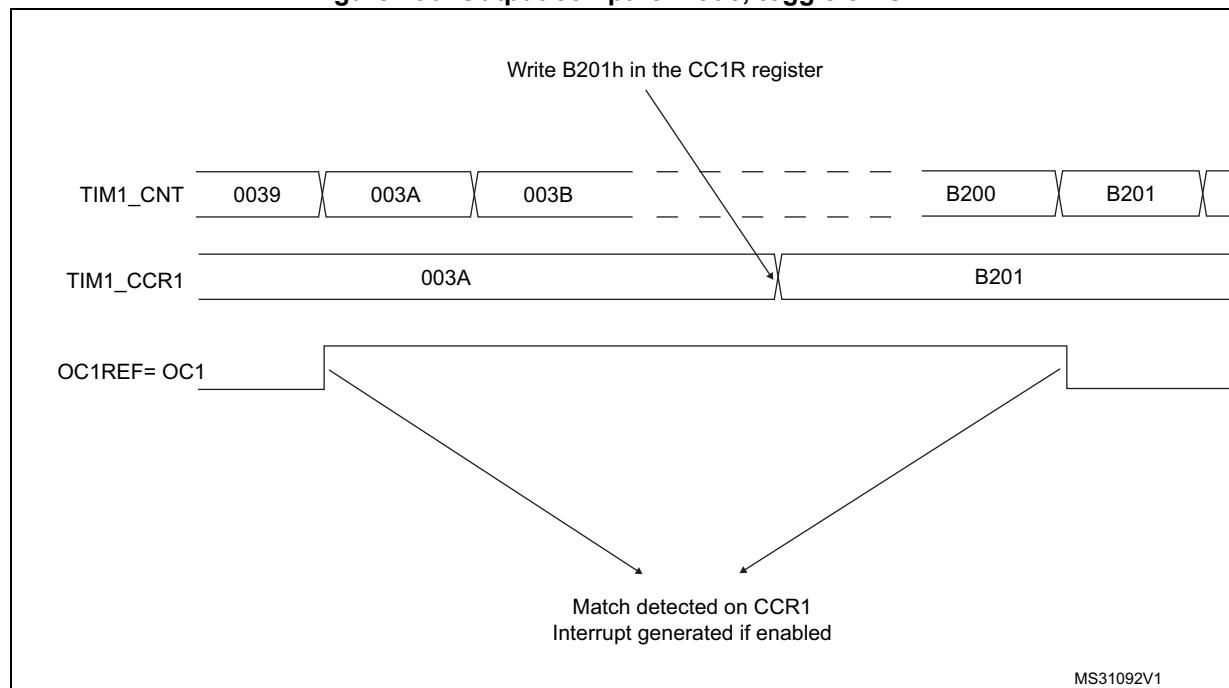
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

### 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 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 196](#).

**Figure 196. Output compare mode, toggle on OC1**



### 18.3.9 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

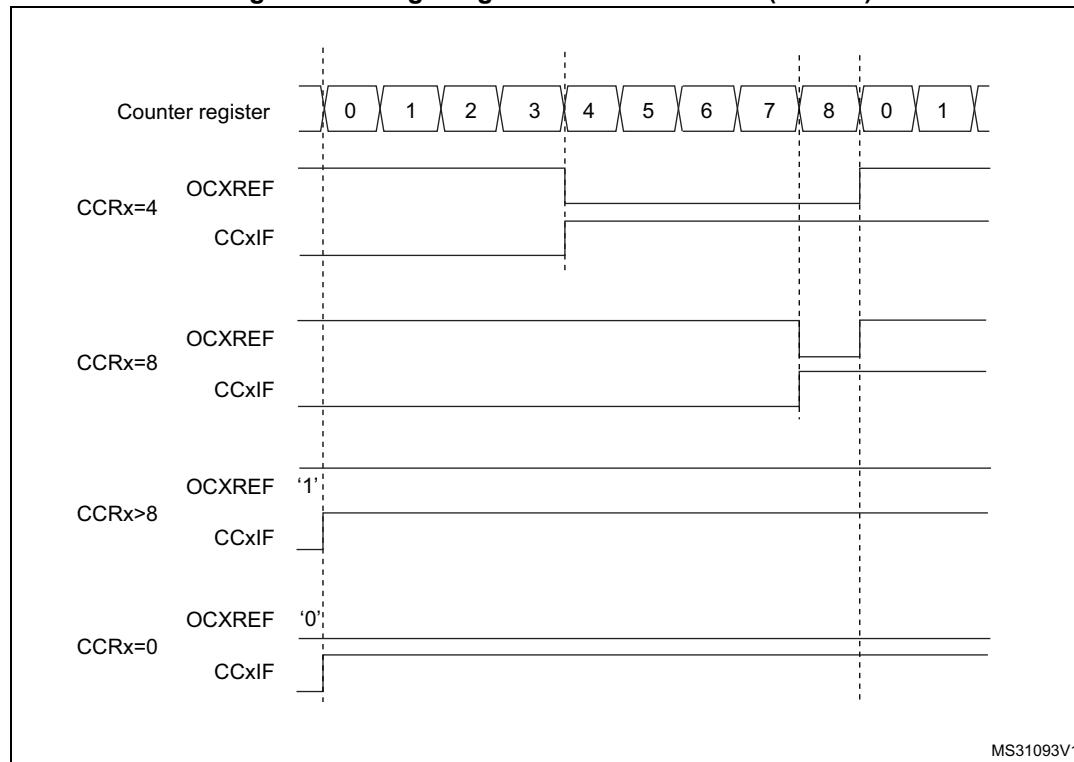
OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSS1 and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description 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 TIM16/TIM17 are capable of upcounting only. Refer to [Upcounting mode on page 497](#).

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 197](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 197. Edge-aligned PWM waveforms (ARR=8)**



### 18.3.10 Complementary outputs and dead-time insertion

The TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OC<sub>x</sub> or complementary OC<sub>xN</sub>) can be selected independently for each output. This is done by writing to the CC<sub>xP</sub> and CC<sub>xNP</sub> bits in the TIM<sub>x</sub>\_CCER register.

The complementary signals OC<sub>x</sub> and OC<sub>xN</sub> are activated by a combination of several control bits: the CC<sub>xE</sub> and CC<sub>xNE</sub> bits in the TIM<sub>x</sub>\_CCER register and the MOE, OIS<sub>x</sub>, OIS<sub>xN</sub>, OSS<sub>I</sub> and OSS<sub>R</sub> bits in the TIM<sub>x</sub>\_BDTR and TIM<sub>x</sub>\_CR2 registers. Refer to [Table 75: Output control bits for complementary OC<sub>x</sub> and OC<sub>xN</sub> channels with break feature \(TIM16/17\) on page 534](#) for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

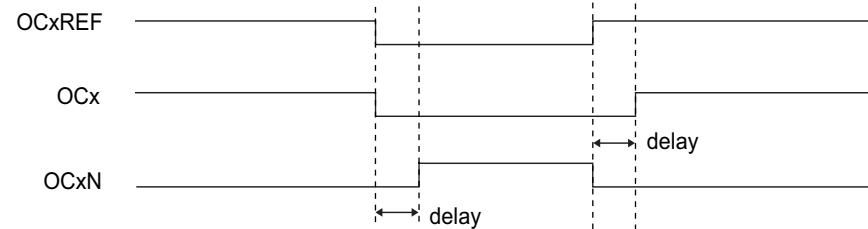
Dead-time insertion is enabled by setting both CC<sub>xE</sub> and CC<sub>xNE</sub> bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OC<sub>xREF</sub>, it generates 2 outputs OC<sub>x</sub> and OC<sub>xN</sub>. If OC<sub>x</sub> and OC<sub>xN</sub> are active high:

- The OC<sub>x</sub> output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OC<sub>xN</sub> output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

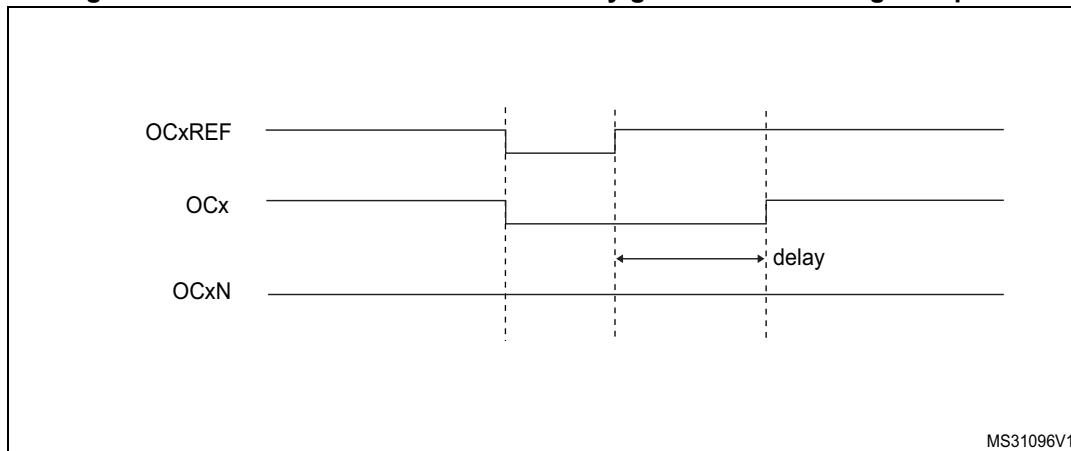
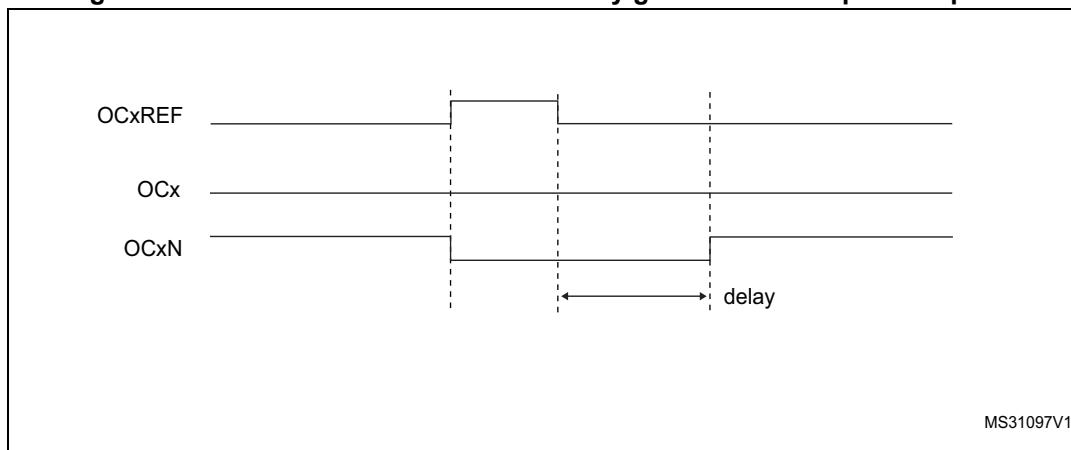
If the delay is greater than the width of the active output (OC<sub>x</sub> or OC<sub>xN</sub>) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OC<sub>xREF</sub>. (we suppose CC<sub>xP</sub>=0, CC<sub>xNP</sub>=0, MOE=1, CC<sub>xE</sub>=1 and CC<sub>xNE</sub>=1 in these examples)

**Figure 198. Complementary output with dead-time insertion.**



MS31095V1

**Figure 199. Dead-time waveforms with delay greater than the negative pulse.****Figure 200. Dead-time waveforms with delay greater than the positive pulse.**

The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx\_BDTR register. Refer to [Section 18.4.14: TIMx break and dead-time register \(TIMx\\_BDTR\)\(x = 16 to 17\) on page 537](#) for delay calculation.

#### Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx\_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

*Note:*

*When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.*

### 18.3.11 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault from input pins. The break circuitry can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx\_BDTR register allows to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSS1 bit in the TIMx\_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx\_CR2 register which are setting the output shutdown level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to [Table 75: Output control bits for complementary OCx and OCxN channels with break feature \(TIM16/17\) on page 534](#) for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx\_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx\_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

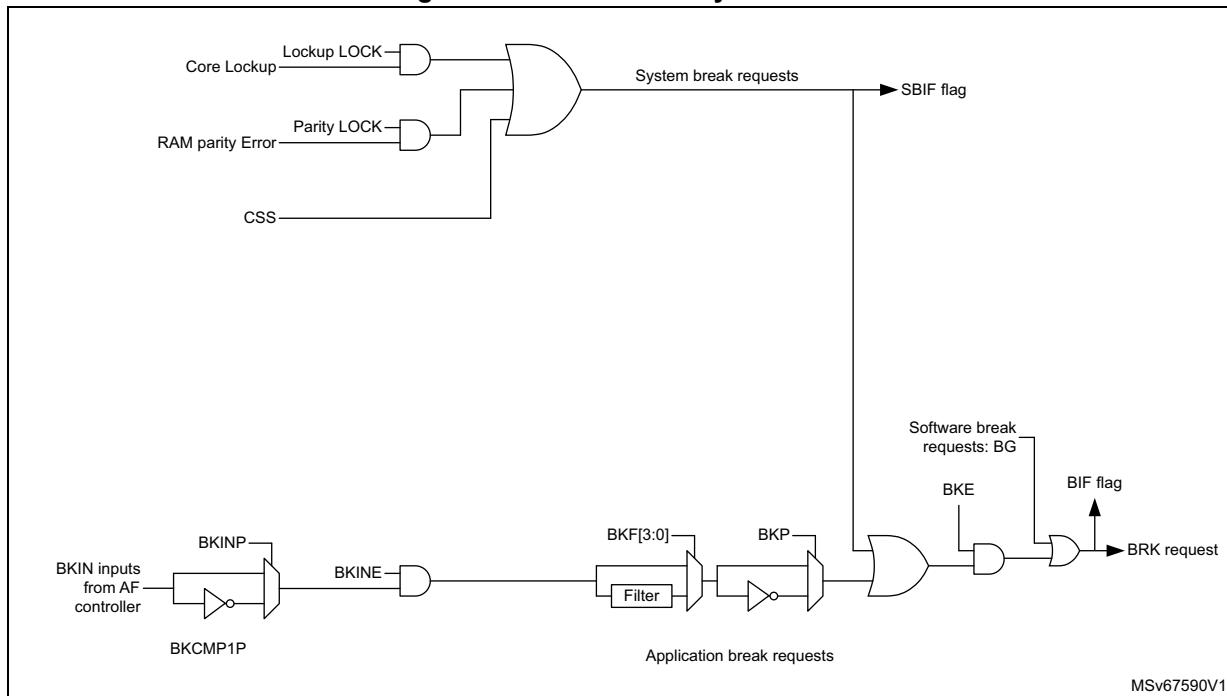
A programmable filter (BKF[3:0] bits in the TIMx\_BDTR register allows to filter out spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx\_AF1 register.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source:
  - A system break:
    - the Cortex®-M0+ LOCKUP output
    - the SRAM parity error signal
    - a clock failure event generated by the CSS detector

Figure 201. Break circuitry overview



**Caution:** An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (example, using the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO (selected by the OSS1 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 OSS1=0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck\_tim clock cycles).
  - If OSS1=0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx\_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance.

Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

- Note:** *If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSS1 value.*  
*If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx\_CR2 register.*

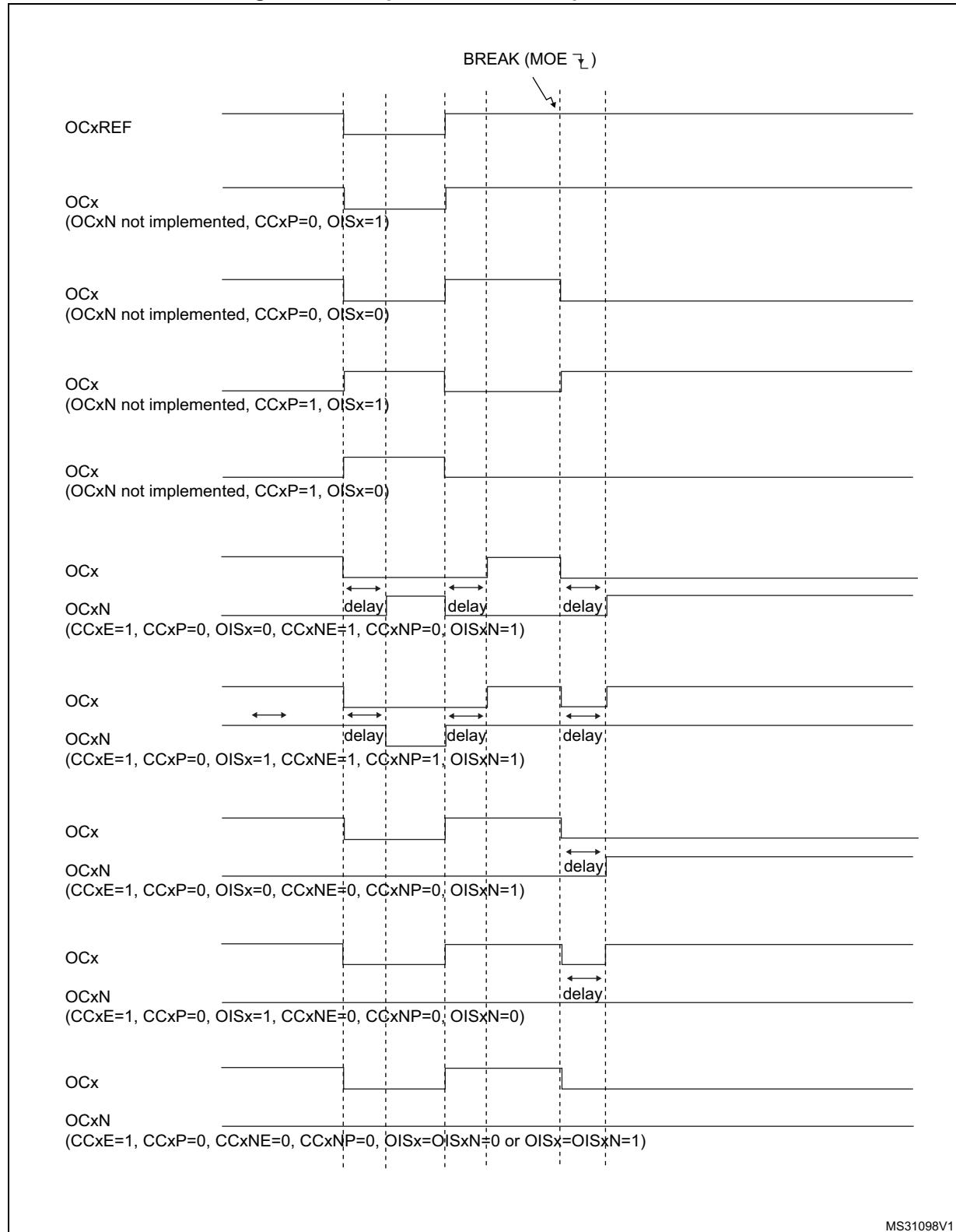
- Note:** *The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.*

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx\_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx\_BDTR register. Refer to [Section 18.4.14: TIMx break and dead-time register \(TIMx\\_BDTR\)\(x = 16 to 17\) on page 537](#). The LOCK bits can be written only once after an MCU reset.

The [Figure 202](#) shows an example of behavior of the outputs in response to a break.

Figure 202. Output behavior in response to a break



MS31098V1

### 18.3.12 Bidirectional break inputs

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on [Figure 203](#).

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see [Table 74](#))

**Table 74. Break protection disarming conditions**

MOE	BKDIR	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

#### Arming and re-arming break circuitry

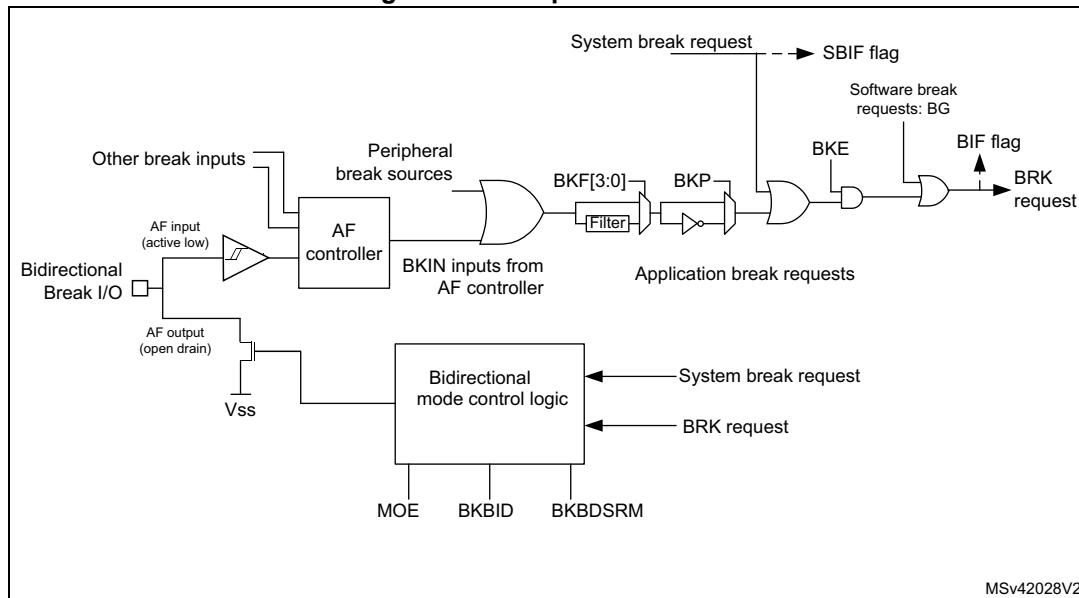
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

**Figure 203. Output redirection**



MSv42028V2

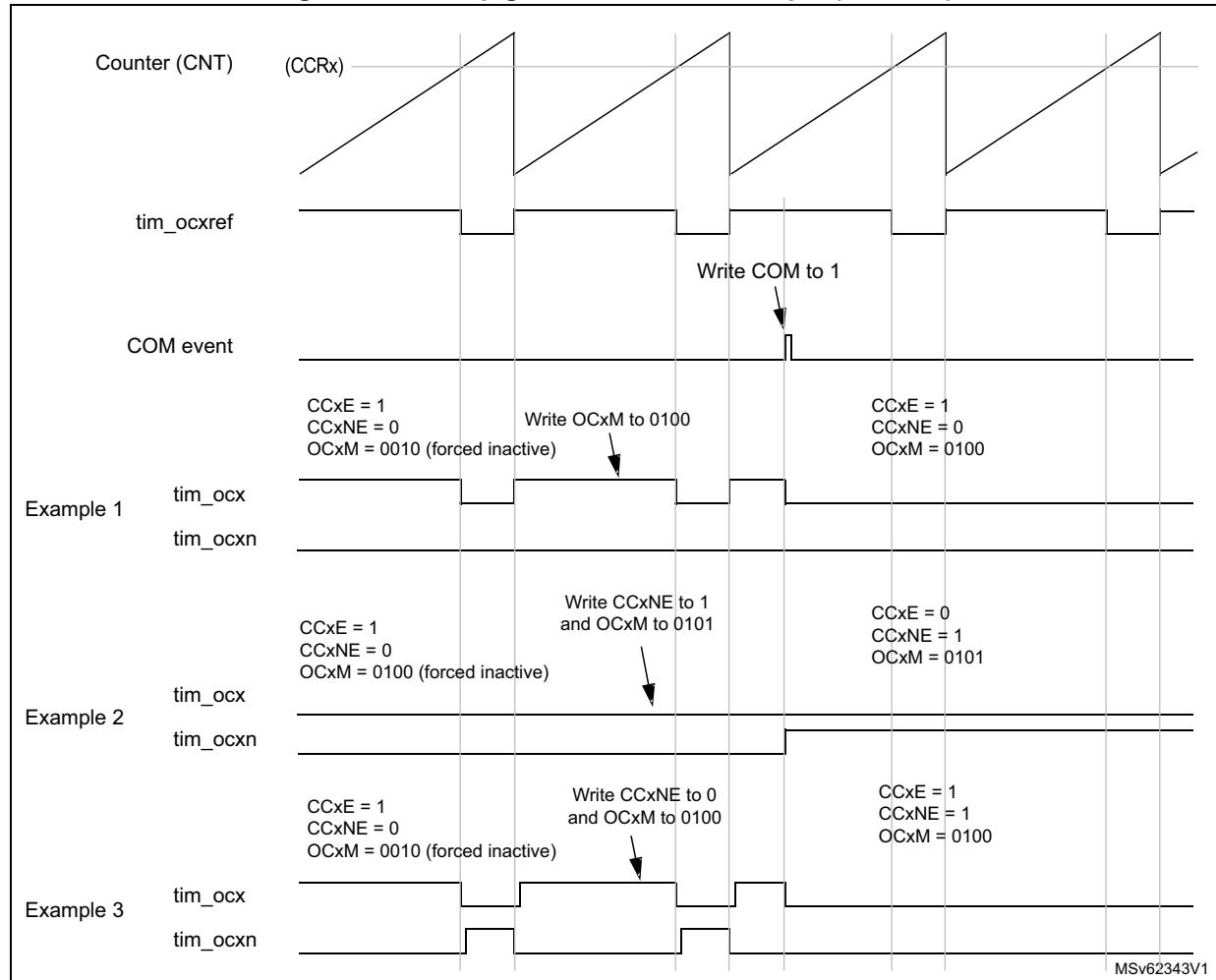
### 18.3.13 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx\_EGR register or by hardware (on tim\_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx\_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx\_DIER register) or a DMA request (if the COMDE bit is set in the TIMx\_DIER register).

The [Figure 204](#) describes the behavior of the tim\_ocx and tim\_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 204. 6-step generation, COM example (OSSR=1)



### 18.3.14 One-pulse mode

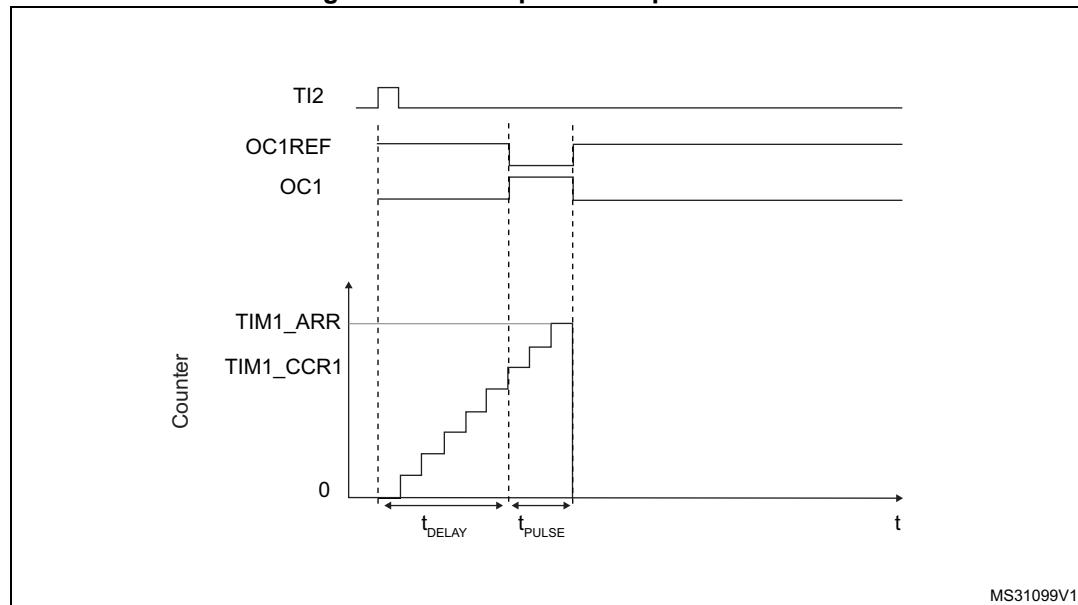
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ )

**Figure 205. Example of one pulse mode**



For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx\_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx\_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx\_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{DELAY}$  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 one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx\_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case one has to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{DELAY}$  min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx\_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 18.3.15 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

### 18.3.16 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

### 18.3.17 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx\_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx\_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx\_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx\_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

00000: TIMx\_CR1,

00001: TIMx\_CR2,

00010: TIMx\_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers ( $x = 2, 3, 4$ ) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:  
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 18.3.18 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx\_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK\_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

### 18.3.19 Debug mode

When the microcontroller enters debug mode (Cortex<sup>®</sup>-M0+ core halted), the TIMx counter either continues to work normally or stops, depending on DBG\_TIMx\_STOP configuration bit in DBG module. For more details, refer to [Section 26.9.2: Debug support for timers, watchdog, and I2C](#).

For safety purposes, when the counter is stopped (DBG\_TIMx\_STOP = 1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

## 18.4 TIM16/TIM17 registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

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

### 18.4.1 TIMx control register 1 (TIMx\_CR1)(x = 16 to 17)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

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

Bit 11 **UIFREMAP**: UIF status bit remapping

- 0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) 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

*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.*

**18.4.2 TIMx control register 2 (TIMx\_CR2)(x = 16 to 17)**

Address offset: 0x04

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.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC

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

**Bit 9 OIS1N:** Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 8 OIS1:** Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

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

**Bit 3 CCDS:** Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

**Bit 2 CCUS:** Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

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.*

### 18.4.3 TIMx DMA/interrupt enable register (TIMx\_DIER)(x = 16 to 17)

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.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE

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

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable

0: CC1 DMA request disabled

1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled

1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable

0: Break interrupt disabled

1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE**: COM interrupt enable

0: COM interrupt disabled

1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

#### 18.4.4 TIMx status register (TIMx\_SR)(x = 16 to 17)

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.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF

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, must be kept at reset value.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx\_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx\_CNT matches the content of the TIMx\_CCR1 register. When the content of TIMx\_CCR1 is greater than the content of TIMx\_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx\_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx\_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx\_EGR register, if URS=0 and UDIS=0 in the TIMx\_CR1 register.

**18.4.5 TIMx event generation register (TIMx\_EGR)(x = 16 to 17)**

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	Res	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, must be kept at reset value.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

### 18.4.6 TIMx capture/compare mode register 1 (TIMx\_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Input capture mode:

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.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]								
								rw	rw	rw	rw	rw	rw	rw	rw

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

#### Bits 7:4 IC1F[3:0]: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{DTS}$
- 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[1:0]: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).

The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input.

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

Others: Reserved

*Note:* CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).

### 18.4.7 TIMx capture/compare mode register 1 [alternate] (TIMx\_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]									
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]									
									rw	rw	rw	rw	rw	rw	rw

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

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

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active.

All other values: Reserved

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

*In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

*The OC1M[3] bit is not contiguous, located in bit 16.*

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/Write operations access the preload register. TIMx\_CCR1 preload value is loaded in the active register at each update event.

*Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S='00' (the channel is configured in output).*

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx\_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

Others: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

### 18.4.8 TIMx capture/compare enable register (TIMx\_CCER)(x = 16 to 17)

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											

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

- 0: OC1N active high
- 1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (the channel is configured in output).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

- 0: Off - OC1N is not active. OC1N level is then function of MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

- 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
- 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

- 0: Capture mode disabled / OC1 is not active (see below)
- 1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 75](#) for details.

**Table 75. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)**

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	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	0		0	0		
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP	
	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		

- When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

**Note:** The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and GPIO control and alternate function registers.

#### 18.4.9 TIMx counter (TIMx\_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.														
r															
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx\_ISR register. If the UIFREMAP bit in TIMx\_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

#### 18.4.10 TIMx prescaler (TIMx\_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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”).

#### 18.4.11 TIMx auto-reload register (TIMx\_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 18.3.1: Time-base unit on page 495](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 18.4.12 TIMx repetition counter register (TIMx\_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP\_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP\_CNT is reloaded with REP value only at the repetition update event U\_RC, any write to the TIMx\_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

### 18.4.13 TIMx capture/compare register 1 (TIMx\_CCR1)(x = 16 to 17)

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).

### 18.4.14 TIMx break and dead-time register (TIMx\_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]
			rw		rw							rw	rw	rw	rw
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 BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID:** Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM:** Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bits 25:20 Reserved, must be kept at reset value.

**Bits 19:16 BKF[3:0]: Break filter**

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously

0001:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , 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

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

**Bit 15 MOE: Main output enable**

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state depending on the OSS1 bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register)

See OC/OCN enable description for more details ([Section 18.4.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 532](#)).

**Bit 14 AOE: Automatic output enable**

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 13 BKP: Break polarity**

0: Break input BRK is active low

1: Break input BRK is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

**Bit 12 BKE: Break enable**

0: Break inputs (BRK and CCS clock failure event) disabled

1: Break inputs (BRK and CCS clock failure event) enabled

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 18.4.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 532](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details ([Section 18.4.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 532](#)).

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> = 2 x t<sub>DTS</sub>

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t<sub>dtg</sub> with t<sub>dtg</sub> = 8 x t<sub>DTS</sub>

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t<sub>dtg</sub> with t<sub>dtg</sub> = 16 x t<sub>DTS</sub>

Example if t<sub>DTS</sub> = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 18.4.15 TIMx DMA control register (TIMx\_DCR)(x = 16 to 17)

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.	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 field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,

00001: 2 transfers,

00010: 3 transfers,

...

10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

00000: TIMx\_CR1,

00001: TIMx\_CR2,

00010: TIMx\_SMCR,

...

**Example:** Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx\_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx\_CR1 address.

### 18.4.16 TIMx DMA address for full transfer (TIMx\_DMAR)(x = 16 to 17)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx\_CR1 address) + (DBA + DMA index) × 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).

### 18.4.17 TIM16 alternate function register 1 (TIM16\_AF1)

Address offset: 0x60

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.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BKINP	Res.	BKINE							
						rw									rw

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

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 18.4.18 TIM16 input selection register (TIM16\_TISEL)

Address offset: 0x68

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.	TI1SEL[3:0]														
															rw

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

- 0000: TIM16\_CH1 input
- 0001: LSI
- 0010: LSE
- 0011: Reserved
- 0100: MCO2
- Others: Reserved

### 18.4.19 TIM17 alternate function register 1 (TIM17\_AF1)

Address offset: 0x60

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.	Res.						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BKINP	Res.	BKINE							
						rw									rw

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

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 18.4.20 TIM17 input selection register (TIM17\_TISEL)

Address offset: 0x68

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.	TI1SEL[3:0]														
															rw

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

- 0000: TIM17\_CH1 input
- 0001: reserved
- 0010: HSE/32
- 0011: MCO
- 0100: MCO2
- Others: Reserved

#### **18.4.21 TIM16/TIM17 register map**

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 76. TIM16/TIM17 register map and reset values**

Table 76. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	TIMx_RCR	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	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.		
0x34	TIMx_CCR1	0	BKBID	0	BKDSTM	0	BKF[3:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		
0x44	TIMx_BDTR	0	BKDSTM	0	BKF[3:0]	0	DBL[4:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		
0x48	TIMx_DCR	0	MOE	0	AOE	0	BKE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		
0x4C	TIMx_DMAR	0	K	0	OSSR	0	OSSI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		
0x60	TIM16_AF1	0	LOC[1:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		
0x60	TIM17_AF1	0	DTG[7:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		
0x68	TIM16_TISEL	0	DMAB[15:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		
0x68	TIM17_TISEL	0	TI1SEL[3:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value	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.		

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 19 Infrared interface (IRTIM)

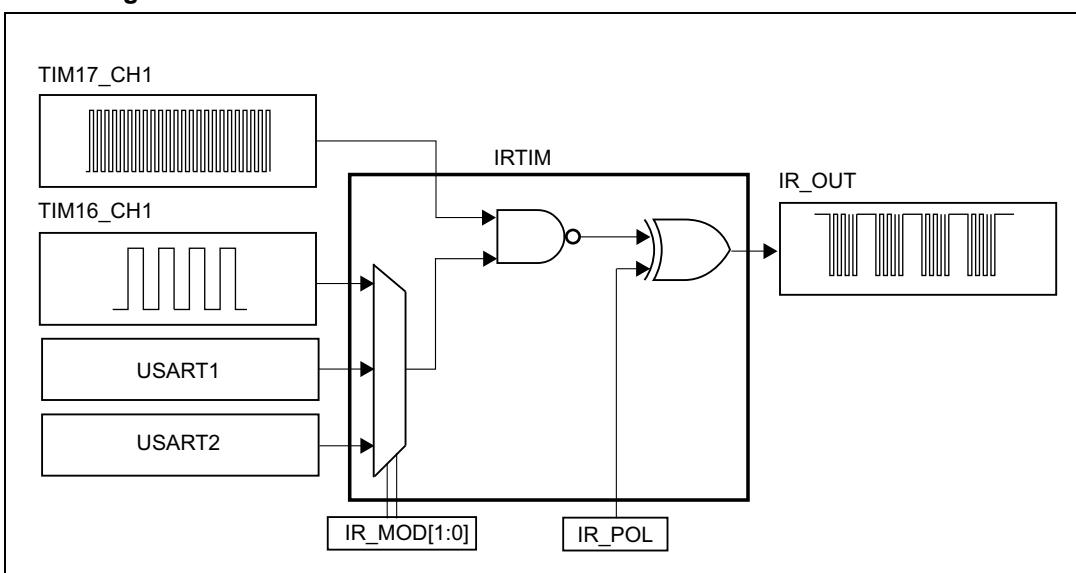
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with USART1, USART2, TIM16 and TIM17 as shown in [Figure 206](#).

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 206. IRTIM 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 or alternatively USART1 or USART2 generates the modulation envelope according to the setting of the IR\_MOD[1:0] bits in the SYSCFG\_CFGR1 register.

The polarity of the output signal from IRTIM is controlled by the IR\_POL bit in the SYSCFG\_CFGR1 register and can be inverted by setting of this bit.

The infrared function is output on the IR\_OUT pin. The activation of this function is done through the GPIOx\_AFRx register by enabling the related alternate function bit.

The high sink LED driver capability (only available on the PB9 and PC14 pins) can be activated through the I2C\_PB9\_FMP bit and/or I2C\_PC14\_FMP bit in the SYSCFG\_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

## 20 Independent watchdog (IWDG)

### 20.1 Introduction

The devices feature an embedded watchdog peripheral that offers a combination of high safety level, timing accuracy and flexibility of use. The Independent watchdog peripheral detects and solves malfunctions due to software failure, and triggers system reset when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to [Section 21: System window watchdog \(WWWDG\)](#).

### 20.2 IWDG main features

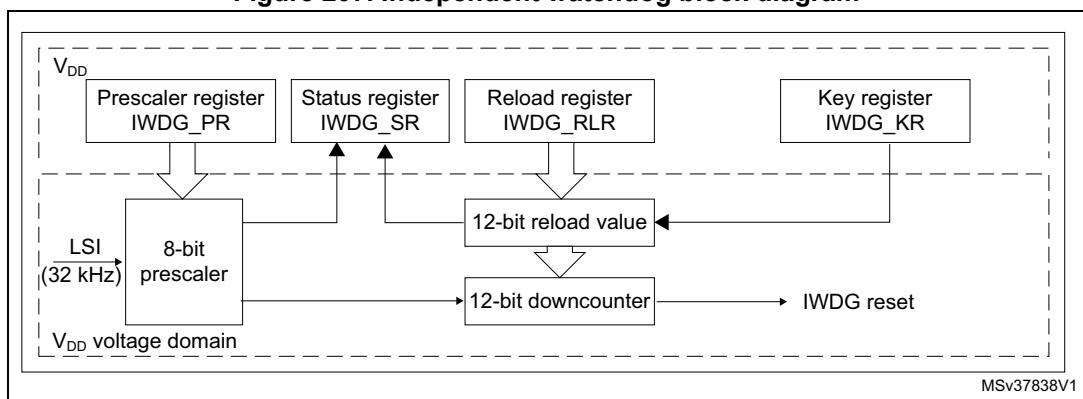
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window

### 20.3 IWDG functional description

#### 20.3.1 IWDG block diagram

*Figure 207* shows the functional blocks of the independent watchdog module.

**Figure 207. Independent watchdog block diagram**



1. The register interface is located in the  $V_{DD}$  voltage domain. The watchdog function is located in the  $V_{DD}$  voltage domain, still functional in Stop and Standby modes.

When the independent watchdog is started by writing the value 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*, the counter starts counting down from the reset value of 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 key register (IWDG\_KR)*, the IWDG\_RLR value is reloaded in the counter and the watchdog reset is prevented.

Once running, the IWDG cannot be stopped.

### 20.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG\_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *IWDG window register (IWDG\_WINR)*, then a reset is provided.

The default value of the *IWDG window register (IWDG\_WINR)* is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed in order to reset the downcounter to the *IWDG reload register (IWDG\_RLR)* value and ease the cycle number calculation to generate the next reload.

#### Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG\_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Write to the *IWDG window register (IWDG\_WINR)*. This automatically refreshes the counter value in the *IWDG reload register (IWDG\_RLR)*.

*Note:* Writing the window value allows the counter value to be refreshed by the RLR when *IWDG status register (IWDG\_SR)* is set to 0x0000 0000.

#### Configuring the IWDG when the window option is disabled

When the window option is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG\_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Refresh the counter value with IWDG\_RLR (IWDG\_KR = 0x0000 AAAA).

### 20.3.3 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the *IWDG key register (IWDG\_KR)* is written by the software before the counter reaches end of count or if the downcounter is reloaded inside the window.

### 20.3.4 Register access protection

Write access to *IWDG prescaler register (IWDG\_PR)*, *IWDG reload register (IWDG\_RLR)* and *IWDG window register (IWDG\_WINR)* is protected. To modify them, the user must first write the code 0x0000 5555 in the *IWDG key register (IWDG\_KR)*. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or of the downcounter reload value or of the window value is ongoing.

### 20.3.5 Debug mode

When the device enters Debug mode (core halted), the IWDG counter either continues to work normally or stops, depending on the configuration of the corresponding bit in DBGMCU freeze register.

## 20.4 IWDG registers

Refer to [Section 1.2 on page 35](#) for a list of abbreviations used in register descriptions.

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

### 20.4.1 IWDG 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 20.3.4: Register access protection](#))

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

## 20.4.2 IWDG 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.														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PR[2:0]														
														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 20.3.4: Register access protection](#). They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the [IWDG status register \(IWDG\\_SR\)](#) must be reset in order to be able to change the prescaler divider.

- 000: divider /4
- 001: divider /8
- 010: divider /16
- 011: divider /32
- 100: divider /64
- 101: divider /128
- 110: divider /256
- 111: divider /256

*Note: Reading this register returns the prescaler value from the V<sub>DD</sub> voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

### 20.4.3 IWDG 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 [Register access protection](#). They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the [IWDG key register \(IWDG\\_KR\)](#). The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to the datasheet for the timeout information.

The RVU bit in the [IWDG status register \(IWDG\\_SR\)](#) must be reset to be able to change the reload value.

*Note: Reading this register returns the reload value from the  $V_{DD}$  voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

## 20.4.4 IWDG 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 five LSI cycles).

Window value can be updated only when WVU bit is reset.

Bit 1 **RVU**: Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V<sub>DD</sub> voltage domain (takes up to five LSI 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 five LSI cycles).

Prescaler value can be updated only when PVU bit is reset.

Note:

*If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.*

## 20.4.5 IWDG 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.

Bits 11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected, see [Section 20.3.4](#), they contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x0

The WVU bit in the [IWDG status register \(IWDG\\_SR\)](#) must be reset in order to be able to change the reload value.

*Note: Reading this register returns the reload value from the  $V_{DD}$  voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

## 20.4.6 IWDG register map

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

**Table 77. IWDG register map and reset values**

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 21 System window watchdog (WWDG)

### 21.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. An MCU reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

### 21.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see [Figure 209](#))
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

### 21.3 WWDG functional description

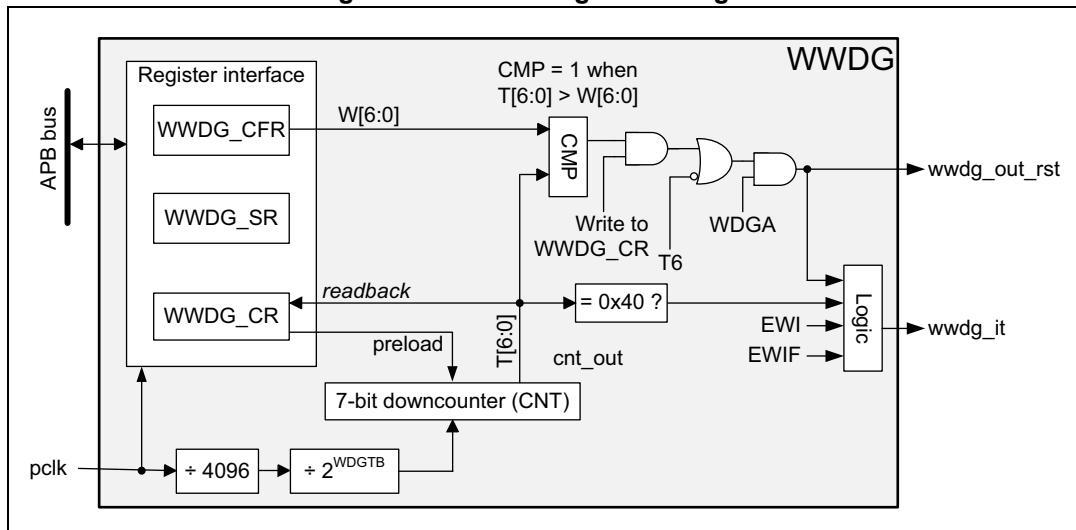
If the watchdog is activated (the WDGA bit is set in the WWDG\_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG\_CR register at regular intervals during normal operation to prevent an MCU reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG\_CR register must be between 0xFF and 0xC0.

Refer to [Figure 208](#) for the WWDG block diagram.

### 21.3.1 WWDG block diagram

Figure 208. Watchdog block diagram



### 21.3.2 Enabling the watchdog

When the user option WWDG\_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG\_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG\_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

### 21.3.3 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG\_CR register (see [Figure 209](#)). The [WWDG configuration register \(WWDG\\_CFR\)](#) contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. [Figure 209](#) describes the window watchdog process.

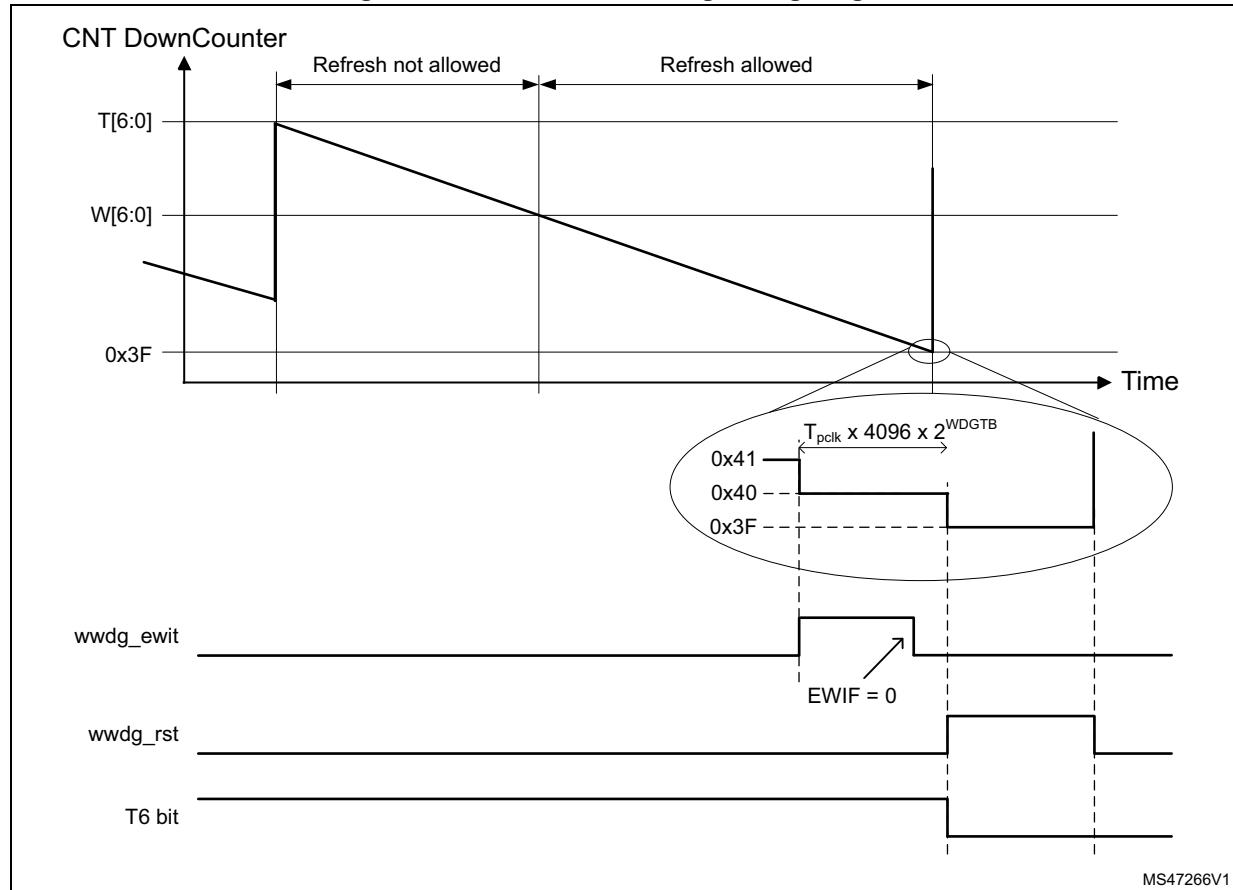
**Note:** The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

### 21.3.4 How to program the watchdog timeout

Use the formula in [Figure 209](#) 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 209. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{\text{WWDG}} = t_{\text{PCLK}} \times 4096 \times 2^{\text{WDGTB}[1:0]} \times (\text{T}[5:0] + 1) \quad (\text{ms})$$

where:

$t_{\text{WWDG}}$ : WWDG timeout

$t_{\text{PCLK}}$ : APB clock period measured in ms

4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[1:0] is set to 3 and T[5:0] is set to 63:

$$t_{\text{WWDG}} = (1 / 48000) \times 4096 \times 2^3 \times (63 + 1) = 43.69\text{ms}$$

Refer to the datasheet for the minimum and maximum values of  $t_{\text{WWDG}}$ .

### 21.3.5 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to [Section 26: Debug support \(DBG\)](#).

## 21.4 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG\_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG\_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing '0' to the EWIF bit in the WWDG\_SR register.

**Note:** *When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.*

## 21.5 WWDG registers

Refer to [Section 1.2 on page 35](#) for a list of abbreviations used in register descriptions.

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

### 21.5.1 WWDG control register (WWDG\_CR)

Address offset: 0x000

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	rw	rw	rw	rw	rw	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, decremented every  $(4096 \times 2^{\text{WDGTB}[1:0]})$  PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

## 21.5.2 WWDG configuration register (WWDG\_CFR)

Address offset: 0x004

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDGTB[2:0]															
Res.	Res.	rw	rw	rw		rs			rw						
W[6:0]															

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

**Bits 13:11 WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

**Bit 10** Reserved, must be kept at reset value.

**Bit 9 EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

**Bits 8:7** Reserved, must be kept at reset value.

**Bits 6:0 W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

### 21.5.3 WWDG status register (WWDG\_SR)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bit 0 **EWIF**: Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

#### **21.5.4 WWDG register map**

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

**Table 78. WWDG register map and reset values**

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 22 Real-time clock (RTC)

### 22.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

### 22.2 RTC main features

The RTC supports the following features (see [Figure 210: RTC block diagram](#)):

- Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- One programmable alarm.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin.

The RTC clock sources can be:

- A 32.768 kHz external crystal (LSE)
- An external resonator or oscillator (LSE)
- The internal low power RC oscillator (LSI, with typical frequency of 32 kHz)
- The high-speed external clock (HSE), divided by a prescaler in the RCC.

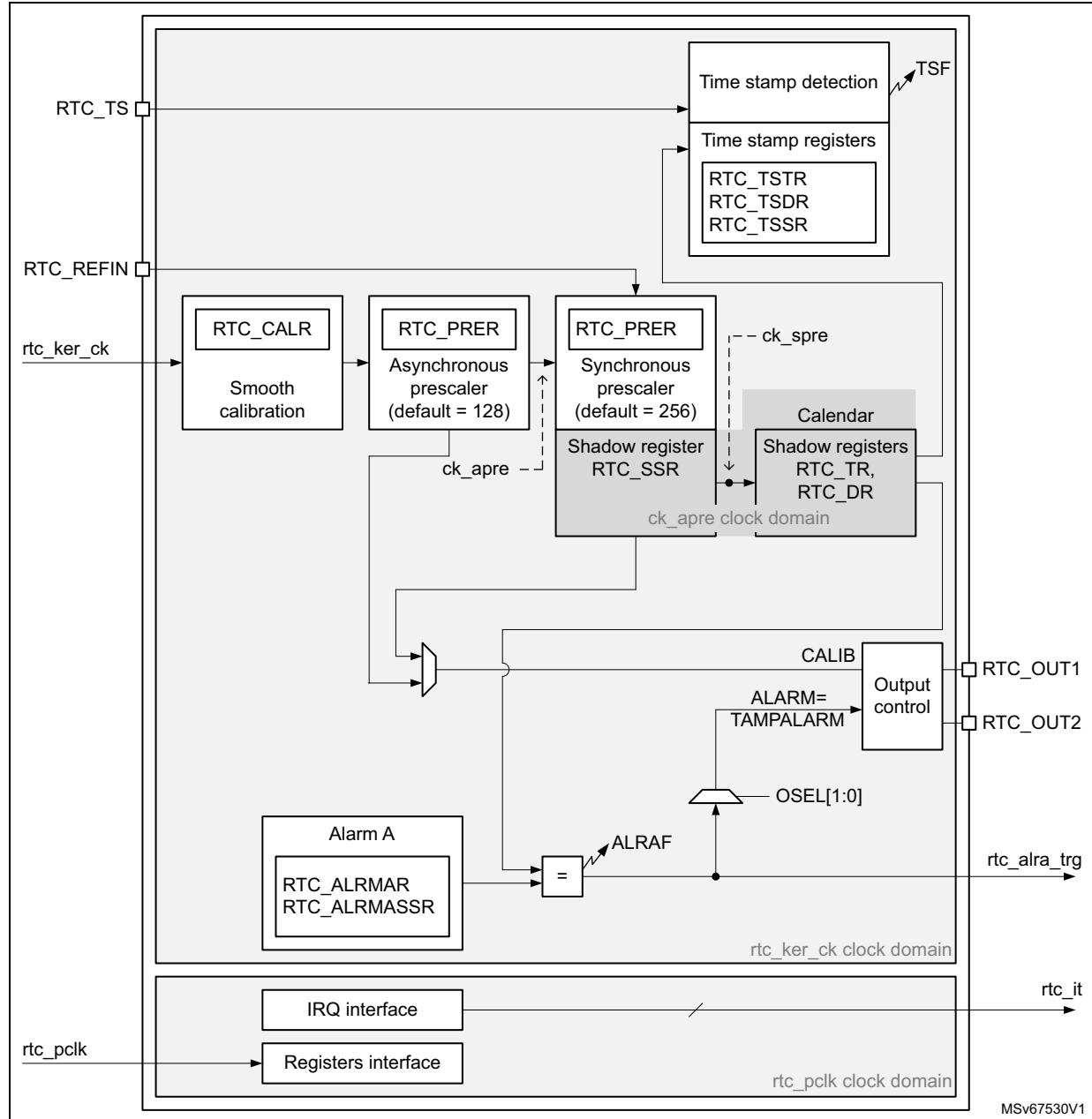
The RTC is functional in all low-power modes except Standby and Shutdown when it is clocked by the LSE or LSI.

All RTC events (Alarm, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.

## 22.3 RTC functional description

### 22.3.1 RTC block diagram

Figure 210. RTC block diagram



### 22.3.2 RTC pins and internal signals

**Table 79. RTC input/output pins**

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

- RTC\_OUT1 and RTC\_OUT2 which selects one of the following two outputs:
  - CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC\_CR register.
  - TAMPALRM: This output is the ALARM output.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC\_CR register which select the alarm A output.

**Table 80. RTC internal input/output signals**

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_it	Output	RTC interrupts (refer to <a href="#">Section 22.5: RTC interrupts</a> for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details).

The triggers outputs can be used as triggers for other peripherals.

### 22.3.3 GPIO controlled by the RTC

RTC\_OUT1 and RTC\_TS are mapped on the same pin.

This pin output mechanism follows the priority order shown in [Table 81](#).

**Table 81. Pin configuration<sup>(1)</sup>**

Pin function		OSEL[1:0] (ALARM output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TSE (RTC_TS input enable)			
TAMPALRM output Push-Pull		01	Don't care	Don't care	0	0	Don't care			
TAMPALRM output Open-Drain <sup>(2)</sup>	No pull	01	Don't care	Don't care	1	0	Don't care			
	Internal pull-up	01	Don't care	Don't care	1	1	Don't care			
CALIB output PP		00	1	0	Don't care	Don't care	Don't care			
RTC_TS input floating		00	0	Don't care	Don't care	Don't care	1			
		00	1	1						
		Don't care	0							
Wakeup pin or Standard GPIO		00	0	Don't care	Don't care	Don't care	0			
		00	1	1						
		Don't care	0							

1. OD: open drain; PP: push-pull.

2. In this configuration the GPIO must be configured in input.

In addition, it is possible to output RTC\_OUT2 thanks to OUT2EN bit. The different functions are mapped on RTC\_OUT1 or on RTC\_OUT2 depending on OSEL, COE and OUT2EN configuration, as show in table [Table 82](#).

**Table 82. RTC\_OUT mapping**

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1	RTC_OUT2
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

### 22.3.4 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 5: Reset and clock control \(RCC\)](#).

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see [Figure 210: 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:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{\text{PREDIV\_A} + 1}$$

The  $ck\_apre$  clock is used to clock the binary RTC\_SSR subseconds downcounter. When it reaches 0, RTC\_SSR is reloaded with the content of PREDIV\_S.

$f_{ck\_spre}$  is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(\text{PREDIV\_S} + 1) \times (\text{PREDIV\_A} + 1)}$$

### 22.3.5 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC\_SSR for the subseconds
- RTC\_TR for the time
- RTC\_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC\_ICSR register is set (see [Section 22.6.9: RTC shift control register \(RTC\\_SHIFTR\)](#)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC\_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC\_SSR, RTC\_TR or RTC\_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock ( $f_{APB}$ ) must be at least 7 times the frequency of the RTC clock ( $f_{RTCCLK}$ ).

The shadow registers are reset by system reset.

### 22.3.6 Programmable alarms

The RTC unit provides programmable alarm: alarm A.

The programmable alarm function is enabled through the ALRAE bit in the RTC\_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC\_ALRMASSR and RTC\_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC\_ALRMAR register, and through the MASKSSx bits of the RTC\_ALRMASSR register.

The alarm interrupt is enabled through the ALRAIE bit in the RTC\_CR register.

**Caution:** If the seconds field is selected (MSK1 bit reset in RTC\_ALRMAR), the synchronous prescaler division factor set in the RTC\_PRER register must be at least 3 to ensure correct behavior.

Alarm A (if enabled by bits OSEL[1:0] in RTC\_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC\_CR register.

### 22.3.7 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 Power-on reset, some of the RTC registers are write-protected.

Writing to the protected RTC registers is enabled by writing a key into the Write Protection register, RTC\_WPR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC\_WPR register.
2. Write 0x53 into the RTC\_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

## Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC\_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC\_ICSR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC\_PRER register.
4. Load the initial time and date values in the shadow registers (RTC\_TR and RTC\_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC\_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note:

*After a system reset, the application can read the INITS flag in the RTC\_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Power-on reset default value (0x00).*

*To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC\_ICSR register.*

## Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC\_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

## Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A.

1. Clear ALRAE in RTC\_CR to disable alarm A.
2. Program the alarm A registers (RTC\_ALRMASSR/RTC\_ALRMAR).
3. Set ALRAE in the RTC\_CR register to enable alarm A again.

**Note:** *Each change of the RTC\_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.*

### 22.3.8 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 ( $f_{RTCCLK}$ ). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC\_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC\_ICSR register each time the calendar registers are copied into the RTC\_SSR, RTC\_TR and RTC\_DR shadow registers. The copy is performed every RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC\_SSR or RTC\_TR locks the values in the higher-order calendar shadow registers until RTC\_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to [Calendar initialization and configuration on page 567](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After synchronization (refer to [Section 22.3.10: 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.

### 22.3.9 Resetting the RTC

The calendar shadow registers (RTC\_SSR, RTC\_TR and RTC\_DR) and some bits of the RTC status register (RTC\_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Power-on 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\_TSDDR) and the alarm A registers (RTC\_ALRMASSR/RTC\_ALRMAR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Power-on reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Power-on reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

### 22.3.10 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 0xFFFF.

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.

### 22.3.11 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\_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck\_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck\_apre period detection window centered on the ck\_spre edge.

When the RTC\_REFIN detection is enabled, PREDIV\_A and PREDIV\_S must be set to their default values:

- PREDIV\_A = 0x007F
- PREDIV\_S = 0x00FF

*Note:* RTC\_REFIN clock detection is not available in Standby mode.

### 22.3.12 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 calibration 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 calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration 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 calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration 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 calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (FCAL) given the input frequency (FRTCCLK) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

### 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 calibration cycle. 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 calibration 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 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

### Re-calibration on-the-fly

The calibration register (RTC\_CALR) can be updated on-the-fly while RTC\_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC\_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC\_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck\_apre cycles after the write operation to RTC\_CALR, the new calibration settings take effect.

### 22.3.13 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC\_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC\_TSSSR, RTC\_TSTR, RTC\_TSDR) when a timestamp event is detected on the RTC\_TS pin.

When a timestamp event occurs, the timestamp flag bit (TSF) in RTC\_SR register is set.

By setting the TSIE bit in the RTC\_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC\_TSTR and RTC\_TSDR) maintain the results of the previous event.

**Note:** *TSF is set 2 ck\_apre cycles after the timestamp event occurs due to synchronization process.*

*There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.*

**Caution:** If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

### 22.3.14 Calibration clock output

When the COE bit is set to 1 in the RTC\_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC\_CR register is reset and PREDIV\_A = 0x7F, the CALIB frequency is  $f_{RTCCLK}/64$ . This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV\_S+1” is a non-zero multiple of 256 (i.e: PREDIV\_S[7:0] = 0xFF), the CALIB frequency is  $f_{RTCCLK}/(256 * (PREDIV_A+1))$ . This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV\_A = 0x7F, PREDIV\_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

**Note:** *When COSEL is cleared, the CALIB output is the output of the 6<sup>th</sup> stage of the asynchronous prescaler.*

*When COSEL is set, the CALIB output is the output of the 8<sup>th</sup> stage of the synchronous prescaler.*

### 22.3.15 Alarm output

The OSEL[1:0] control bits in the RTC\_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flag in the RTC\_SR register.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC\_CR so that the opposite of the selected flags bit is output when POL is set to 1.

#### TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM\_TYPE in the RTC\_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM\_PU in the RTC\_CR.

**Note:** *Once the TAMPALRM output is enabled, it has priority over CALIB on RTC\_OUT1.*

*In case the TAMPALRM is configured open-drain in the RTC, the RTC\_OUT1 GPIO must be configured as input.*

## 22.4 RTC low-power modes

**Table 83. 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 interrupts cause the device to exit the Stop mode.
Standby	The RTC is powered down and must be re-initialized after exiting Standby mode.
Shutdown	The RTC is powered down and must be re-initialized after exiting Shutdown mode.

The table below summarizes the RTC pins and functions capability in all modes.

**Table 84. RTC pins functionality over modes**

Functions	Functional in all low-power modes except Standby and Shutdown modes	Functional in Standby and Shutdown mode
RTC_TS	Yes	No
RTC_REFIN	Yes	No
RTC_OUT1	Yes	No
RTC_OUT2	Yes	No

## 22.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

**Table 85. Interrupt requests**

Interrupt acronym	Interrupt event	Event flag <sup>(1)</sup>	Enable control bit <sup>(2)</sup>	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode	Exit from Standby and Shutdown mode
RTC	Alarm A	ALRAF	ALRAIE	write 1 in CALRAF	Yes	Yes <sup>(3)</sup>	No
	Timestamp	TSF	TSIE	write 1 in CTSF	Yes	Yes <sup>(3)</sup>	No

1. The event flags are in the RTC\_SR register.

2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC\_MISR register.

3. Wakeup from Stop mode is possible only when the RTC clock source is LSE or LSI.

## 22.6 RTC registers

Refer to [Section 1.2 on page 35](#) of the reference manual for a list of abbreviations used in register descriptions.

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

### 22.6.1 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 567](#) and [Reading the calendar on page 568](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x00

Power-on reset value: 0x0000 0000

System reset value: 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	<b>HT[1:0]</b>				<b>HU[3:0]</b>	
										RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	<b>MNT[2:0]</b>				<b>MNU[3:0]</b>				Res.	<b>ST[2:0]</b>				<b>SU[3:0]</b>	
	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

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

## 22.6.2 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 567](#) and [Reading the calendar on page 568](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x04

Power-on reset value: 0x0000 2101

System reset value: 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[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

**Note:** *The calendar is frozen when reaching the maximum value, and can't roll over.*

### 22.6.3 RTC sub second register (RTC\_SSR)

Address offset: 0x08

Power-on reset value: 0x0000 0000

System reset value: 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.
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[15:0]**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

$$\text{Second fraction} = (\text{PREDIV\_S} - \text{SS}) / (\text{PREDIV\_S} + 1)$$

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.

### 22.6.4 RTC initialization control and status register (RTC\_ICSR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x0C

Power-on reset value: 0x0000 0007

System reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 000x xxxx (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.	RECALPF									
															r
Res.	INIT	INITF	RSF	INITS	SHPF	Res.	Res.	ALRAWF							
								rw	r	rc_w0	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](#).

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

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC\_TR and RTC\_DR), and prescaler register (RTC\_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC\_SSR, RTC\_TR and RTC\_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Power-on reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC\_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bits 2:1 Reserved, must be kept at reset value.

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

## 22.6.5 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 567](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x10

Power-on 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:

$$\text{ck_apre frequency} = \text{RTCCLK frequency}/(\text{PREDIV\_A}+1)$$

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV\_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:

$$\text{ck_spre frequency} = \text{ck_apre frequency}/(\text{PREDIV\_S}+1)$$

## 22.6.6 RTC control register (RTC\_CR)

*This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).*

Address offset: 0x18

Power-on reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2 EN	TAMP ALRM_ TYPE	TAMP ALRM_ PU	Res.	Res.	Res.	Res.	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw						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	Res.	Res.	ALRA IE	TSE	Res.	Res.	ALRAE	Res.	FMT	BYP SHAD	REFCK ON	TS EDGE	Res.	Res.	Res.
rw			rw	rw			rw		rw	rw	rw	rw			

Bit 31 **OUT2EN**: RTC\_OUT2 output enable

Setting this bit allows to remap the RTC outputs on RTC\_OUT2 as follows:

**OUT2EN = 0**: RTC output 2 disable

If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC\_OUT1

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC\_OUT1

**OUT2EN = 1**: RTC output 2 enable

If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC\_OUT2

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC\_OUT2

If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC\_OUT2 and TAMPALRM is output on RTC\_OUT1.

Bit 30 **TAMPALRM\_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output

1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM\_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output

1: A pull-up is applied on TAMPALRM output

Bits 28:24 Reserved, must be kept at reset value.

Bit 23 **COE**: Calibration output enable

This bit enables the CALIB output

0: Calibration output disabled

1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to TAMPALRM output.

00: Output disabled

01: Alarm A output enabled

10: Reserved

11: Reserved

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of TAMPALRM output.

0: The pin is high when ALRAF is asserted (depending on OSEL[1:0]).

1: The pin is low when ALRAF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection

When COE = 1, this bit selects which signal is output on CALIB.

0: Calibration output is 512 Hz

1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV\_A = 127 and PREDIV\_S = 255). Refer to [Section 22.3.14: Calibration clock output](#).

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp interrupt disable

1: Timestamp interrupt enable

Bits 14:13 Reserved, must be kept at reset value.

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

Bits 10:9 Reserved, must be kept at reset value.

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 **BYPSHAD**: 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**: Timestamp event active edge

0: RTC\_TS input rising edge generates a timestamp event

1: RTC\_TS input falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 Reserved, must be kept at reset value.

*Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC\_ICSR/INITF = 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.*

## 22.6.7 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[7:0]							
								w	w	w	w	w	w	w	w

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to [RTC register write protection](#) for a description of how to unlock RTC register write protection.

## 22.6.8 RTC calibration register (RTC\_CALR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x28

Power-on 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							

Bits 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:  $(512 \times \text{CALP}) - \text{CALM}$ .

Refer to [Section 22.3.12: 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 22.3.12: 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 22.3.12: 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 22.3.12: RTC smooth digital calibration on page 570](#).

## 22.6.9 RTC shift control register (RTC\_SHIFTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x2C

Power-on 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.	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	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\_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC\_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (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:

Advance (seconds) = (1 - (SUBFS / (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.*

## 22.6.10 RTC timestamp time register (RTC\_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC\_SR. It is cleared when TSF bit is reset.

Address offset: 0x30

Power-on 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	<b>HT[1:0]</b>			<b>HU[3:0]</b>		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	<b>MNT[2:0]</b>			<b>MNU[3:0]</b>			Res.	<b>ST[2:0]</b>			<b>SU[3:0]</b>				
	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.

### 22.6.11 RTC timestamp date register (RTC\_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC\_SR. It is cleared when TSF bit is reset.

Address offset: 0x34

Power-on 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[2:0]				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[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

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

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

### 22.6.12 RTC timestamp sub second register (RTC\_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC\_SR. It is cleared when the TSF bit is reset.

Address offset: 0x38

Power-on 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[15:0]: Sub second value

SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

### 22.6.13 RTC alarm A register (RTC\_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC\_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x40

Power-on 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	WD SEL	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.

### 22.6.14 RTC alarm A sub second register (RTC\_ALRMASSR)

This register can be written only when ALRAWF is set to 1 in RTC\_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 567](#).

Address offset: 0x44

Power-on 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
SS[14:0]															
Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	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.

*Note: The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.*

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

### 22.6.15 RTC status register (RTC\_SR)

Address offset: 0x50

Power-on 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.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TSOVF	TSF	Res.	Res.	ALRAF										
											r	r			r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC\_TR and RTC\_DR) match the alarm A register (RTC\_ALRMAR).

**Note:** *The bits of this register are cleared 2 APB clock cycles after setting their corresponding clear bit in the RTC\_SCR register.*

## 22.6.16 RTC masked interrupt status register (RTC\_MISR)

Address offset: 0x54

Power-on 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.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TSOVMF	TSMF	Res.	Res.	ALRAMF										
											r	r			r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TSOVMF**: Timestamp overflow masked flag

This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp masked flag

This flag is set by hardware when a timestamp interrupt occurs.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **ALRAMF**: Alarm A masked flag

This flag is set by hardware when the alarm A interrupt occurs.

## 22.6.17 RTC status clear register (RTC\_SCR)

Address offset: 0x5C

Power-on 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.											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CTSOVF	CTS F	Res.	Res.	CALRAF										
											w	w			w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CTSOVF**: Clear timestamp overflow flag

Writing 1 in this bit clears the TSOVF bit in the RTC\_SR register.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag

Writing 1 in this bit clears the TSOVF bit in the RTC\_SR register.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **CALRAF**: Clear alarm A flag

Writing 1 in this bit clears the ALRAF bit in the RTC\_SR register.

## 22.6.18 RTC register map

Table 86. RTC register map and reset values

Offset	Register	Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x00	RTC_TR		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	RTC_DR			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																																			
0x08	RTC_SSR				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																																			
0x0C	RTC_ICSR					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																																			
0x10	RTC_PRER																																			
	Reset value																																			
0x14	Reserved																																			
0x18	RTC_CR		OUT2EN																																	
	Reset value	0	0	TAMPALRM_TYPE																																1
0x20	Reserved																																			
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.	Res.	Res.					
	Reset value																																			
0x28	RTC_CALR					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																																			
0x2C	RTC_SHIFTR		ADD1S			PM																														
	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					
0x30	RTC_TSTR		Res.			HT[1:0]																														
	Reset value																																			
0x34	RTC_TSDR		Res.			WDU[1:0]																														
	Reset value																																			
0x38	RTC_TSSSR		Res.																																	
	Reset value																																			

**Table 86. 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
0x40	<b>RTC_ALRMAR</b>	MSK4	WDSEL	DT [1:0]	DU[3:0]	MSK3	PM	HT [1:0]	HU[3:0]	MSK2	MNT[2:0]	MNU[3:0]	SS[14:0]														SU[3:0]						
		0	0	0	0	0	0	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	<b>RTC_ALRMASSR</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.		
0x48 - 0x4C	Reserved																																
0x50	<b>RTC_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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
0x54	<b>RTC_MISR</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.		
0x58	Reserved																																
0x5C	<b>RTC_SCR</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.		

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 23 Inter-integrated circuit (I2C) interface

### 23.1 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.

### 23.2 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 23.3](#)):

- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock: a choice of independent clock sources allowing the I<sup>2</sup>C communication speed to be independent from the PCLK reprogramming

- Wake-up from Stop mode on address match

### 23.3 I<sup>2</sup>C implementation

The devices incorporate one I<sup>2</sup>C-bus controller with the list of features shown in the following table

**Table 87. STM32C0x1 I<sup>2</sup>C implementation**

I <sup>2</sup> C features <sup>(1)</sup>	I <sup>2</sup> C1
7-bit addressing mode	X
10-bit addressing mode	X
Standard mode (up to 100 kbit/s)	X
Fast mode (up to 400 kbit/s)	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X
Independent clock	X
Wake-up from Stop mode	X
SMBus/PMbus	X

1. X = supported.

### 23.4 I<sup>2</sup>C functional description

In addition to receiving and transmitting data, this interface converts them 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.

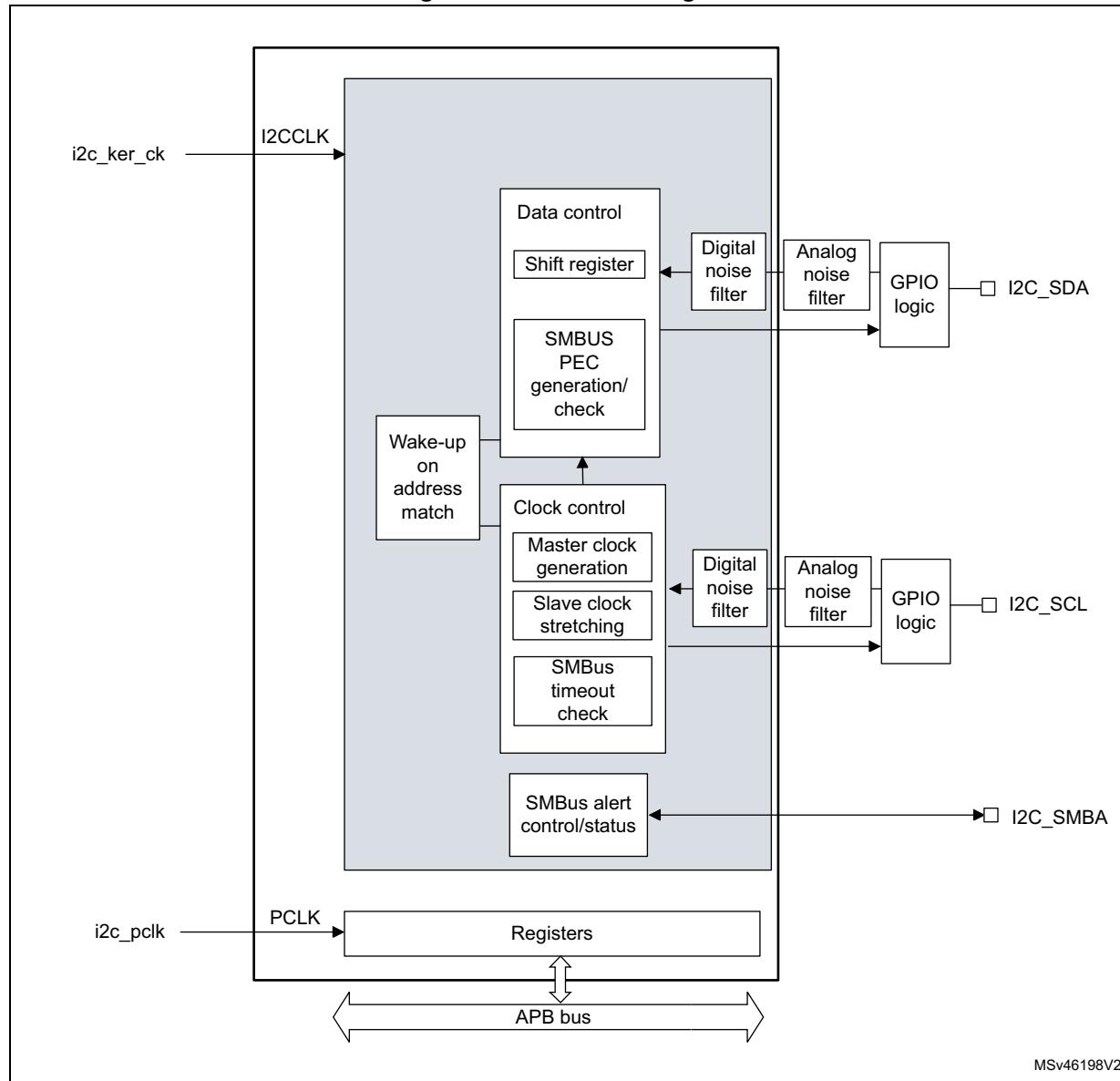
This interface can also be connected to an SMBus with data (SDA) and clock (SCL) pins.

If the SMBus feature is supported, the optional SMBus Alert pin (SMBA) is also available.

### 23.4.1 I2C block diagram

The block diagram of the I2C interface is shown in [Figure 211](#).

**Figure 211. I2C block diagram**



The I<sub>2</sub>C is clocked by an independent clock source, which allows the I<sub>2</sub>C to operate independently from the PCLK frequency.

For I<sub>2</sub>C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). Refer to [Section 23.3: I<sub>2</sub>C implementation](#).

### 23.4.2 I2C pins and internal signals

**Table 88. I2C input/output pins**

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I2C data
I2C_SCL	Bidirectional	I2C clock
I2C_SMBA	Bidirectional	SMBus alert

**Table 89. I2C internal input/output signals**

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to <a href="#">Table 103</a> for the full list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

### 23.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period  $t_{I2CCLK}$  must respect the following conditions:

- $t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$
- $t_{I2CCLK} < t_{HIGH}$

with:

$t_{LOW}$ : SCL low time and  $t_{HIGH}$ : SCL high time

$t_{filters}$ : when enabled, sum of the delays brought by the analog and by the digital filters.

The digital filter delay is DNF  $\times t_{I2CCLK}$ .

The PCLK clock period  $t_{PCLK}$  must respect the condition:

- $t_{PCLK} < 4 / 3 t_{SCL}$  ( $t_{SCL}$ : SCL period)

**Caution:** When the I2C kernel is clocked by PCLK, this clock must respect the conditions for  $t_{I2CCLK}$ .

### 23.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

### Communication flow

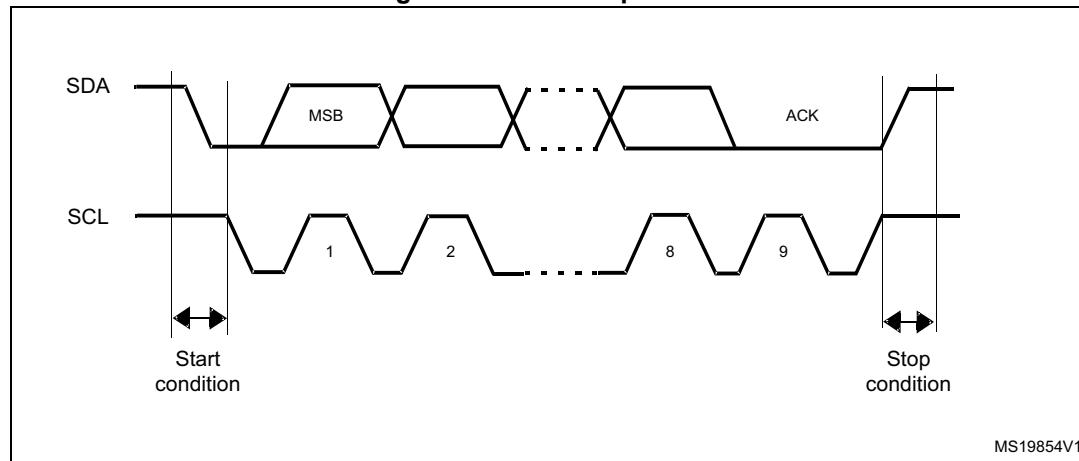
In master mode, the I<sup>2</sup>C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

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 contains the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in master mode.

A ninth clock pulse follows the eight clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter (see [Figure 212](#)).

**Figure 212. I<sup>2</sup>C bus protocol**



Acknowledge can be enabled or disabled by software. The I<sup>2</sup>C interface addresses can be selected by software.

### 23.4.5 I<sup>2</sup>C initialization

#### Enabling and disabling the peripheral

The I<sup>2</sup>C peripheral clock must be configured and enabled in the clock controller, then the I<sup>2</sup>C can be enabled by setting the PE bit in the I<sup>2</sup>C\_CR1 register.

When the I<sup>2</sup>C is disabled (PE = 0), the I<sup>2</sup>C performs a software reset. Refer to [Section 23.4.6](#) for more details.

#### Noise filters

Before enabling the I<sup>2</sup>C peripheral by setting the PE bit in I<sup>2</sup>C\_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This filter is compliant with the I<sup>2</sup>C specification, which requires the suppression of spikes with pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The

user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C\_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than  $DNF \times I2CCLK$  periods. This allows spikes with a programmable length of 1 to 15 I2CCLK periods to be suppressed.

**Table 90. Comparison of analog vs. digital filters**

-	Analog filter	Digital filter
Pulse width of suppressed spikes	$\geq 50$ ns	Programmable length, from one to fifteen I2C peripheral clocks
Benefits	Available in Stop mode	<ul style="list-style-type: none"> <li>– Programmable length: extra filtering capability versus standard requirements</li> <li>– Stable length</li> </ul>
Drawbacks	Variation vs. temperature, voltage, process	Wake-up from Stop mode on address match is not available when digital filter is enabled

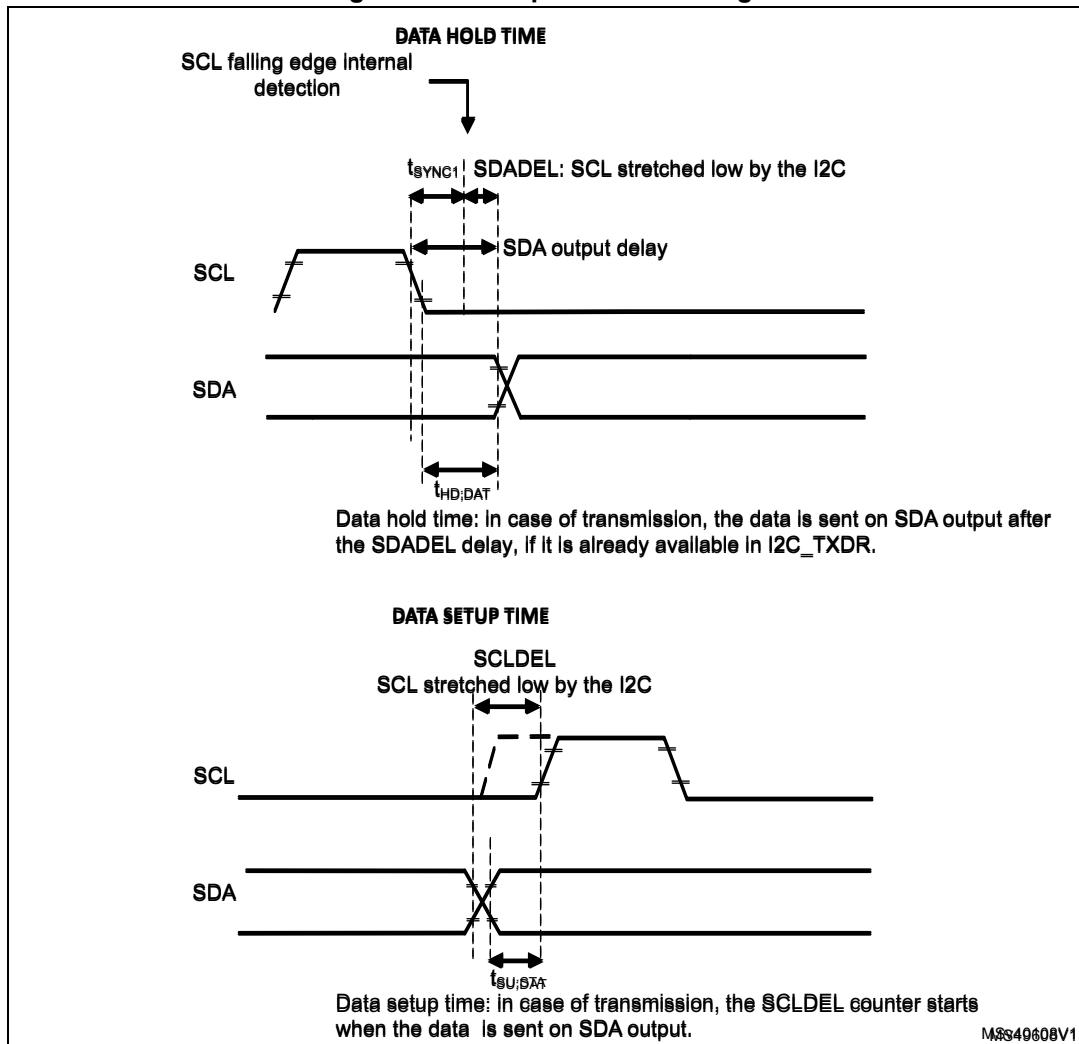
**Caution:** The filter configuration cannot be changed when the I2C is enabled.

### I2C timings

The timings must be configured to guarantee correct data hold and setup times, in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C\_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C configuration window.

Figure 213. Setup and hold timings



When the SCL falling edge is internally detected, a delay ( $t_{SDADEL}$ , impacting the hold time  $t_{HD;DAT}$ ) is inserted before sending SDA output:  $t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}$ , where  $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$ .

The total SDA output delay is:

$$t_{SYNC1} + \{[SDADEL \times (PRESC + 1) + 1] \times t_{I2CCLK}\}$$

$t_{SYNC1}$  duration depends upon:

- SCL falling slope
- When enabled, input delay brought by the analog filter:  $t_{AF(min)} < t_{AF} < t_{AF(max)}$
- When enabled, input delay brought by the digital filter:  $t_{DNF} = DNF \times t_{I2CCLK}$
- Delay due to SCL synchronization to I2CCLK clock (two to three I2CCLK periods)

To bridge the undefined region of the SCL falling edge, the user must program SDADEL in such a way that:

$$\{t_f(\max) + t_{HD;DAT}(\min) - t_{AF(\min)} - [(DNF + 3) \times t_{I2CCLK}] / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$$

$$SDADEL \leq \{t_{HD;DAT}(\max) - t_{AF(\max)} - [(DNF + 4) \times t_{I2CCLK}] / \{(PRESC + 1) \times t_{I2CCLK}\}$$

**Note:**  $t_{AF(min)} / t_{AF(max)}$  are part of the equation only when the analog filter is enabled. Refer to the device datasheet for  $t_{AF}$  values.

The maximum  $t_{HD;DAT}$  can be, respectively, 3.45, 0.9, and 0.45  $\mu s$  for Standard-mode, Fast-mode, and Fast-mode Plus. It must be lower than the maximum of  $t_{VD;DAT}$  by a transition time. This maximum must only be met if the device does not stretch the LOW period ( $t_{LOW}$ ) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. In this case the previous equation becomes:

$$SDADEL \leq \{t_{VD;DAT}(\max) - t_r(\max) - t_{AF}(\max) - [(DNF + 4) \times t_{I2CCLK}] \} / \{(PRESC + 1) \times t_{I2CCLK}\}.$$

**Note:** This condition can be violated when  $NOSTRETCH = 0$ , because the device stretches SCL low to guarantee the set-up time, according to the  $SCLDEL$  value.

Refer to [Table 91](#) for  $t_f$ ,  $t_r$ ,  $t_{HD;DAT}$ , and  $t_{VD;DAT}$  standard values.

- After  $t_{SDADEL}$ , or after sending SDA output when the slave had to stretch the clock because the data was not yet written in I2C\_TXDR register, SCL line is kept at low level during the setup time. This setup time is  $t_{SCLDEL} = (SCLDEL+1) \times t_{PRESC}$ , where  $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$ .  $t_{SCLDEL}$  impacts the setup time  $t_{SU;DAT}$ .

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program  $SCLDEL$  in such a way that:

$$\{[t_r(\max) + t_{SU;DAT}(\min)] / [(PRESC+1) \times t_{I2CCLK}] \} - 1 \leq SCLDEL$$

Refer to [Table 91](#) for  $t_r$  and  $t_{SU;DAT}$  standard values.

The SDA and SCL transition time values to use 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.

**Note:** At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least  $[(SDADEL + SCLDEL + 1) \times (PRESC + 1) + 1] \times t_{I2CCLK}$ , in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C\_TXDR when  $SDADEL$  counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and  $SCLDEL$  counter starts, continuing stretching SCL low to guarantee the data setup time.

If  $NOSTRETCH = 1$  in slave mode, the SCL is not stretched. Consequently the  $SDADEL$  must be programmed so that it guarantees a sufficient setup time.

**Table 91. I<sup>2</sup>C-SMBus specification data setup and hold times**

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	<b>Data hold time</b>	0	-	0	-	0	-	0.3	-	$\mu s$
$t_{VD;DAT}$	<b>Data valid time</b>	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	<b>Data setup time</b>	250	-	100	-	50	-	250	-	
$t_r$	<b>Rise time of both SDA and SCL signals</b>	-	1000	-	300	-	120	-	1000	$ns$
$t_f$	<b>Fall time of both SDA and SCL signals</b>	-	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] bit fields in the I2C\_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output.  
This delay is  $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$  where  $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$ .  
 $t_{SCLL}$  impacts the SCL low time  $t_{LOW}$ .
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is  $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$ , where  $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$ .  $t_{SCLH}$  impacts the SCL high time  $t_{HIGH}$ .

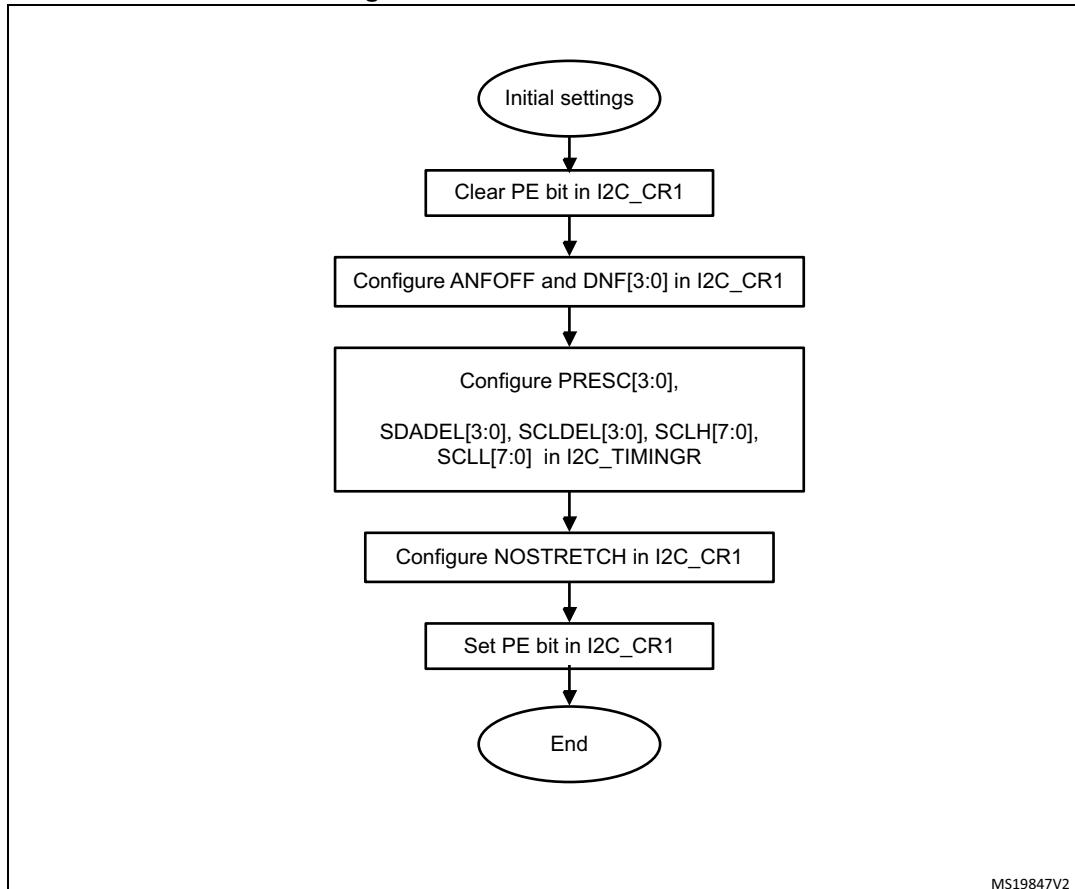
Refer to [I2C master initialization](#) for more details.

**Caution:** Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral.  
Refer to [I2C slave initialization](#) for more details.

**Caution:** Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

**Figure 214. I2C initialization flow**



MS19847V2

### 23.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C\_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and

communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C\_CR2 register: START, STOP, NACK
2. I2C\_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C\_CR2 register: PECBYTE
2. I2C\_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least three APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

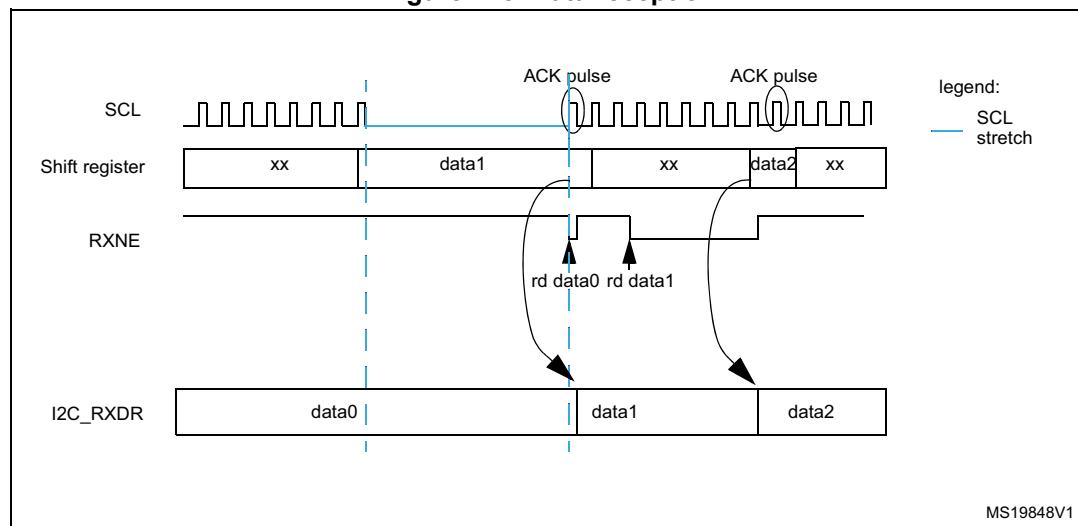
### 23.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

#### Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into I2C\_RXDR register if it is empty (RXNE = 0). If RXNE = 1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C\_RXDR is read. The stretch is inserted between the eighth and ninth SCL pulse (before the acknowledge pulse).

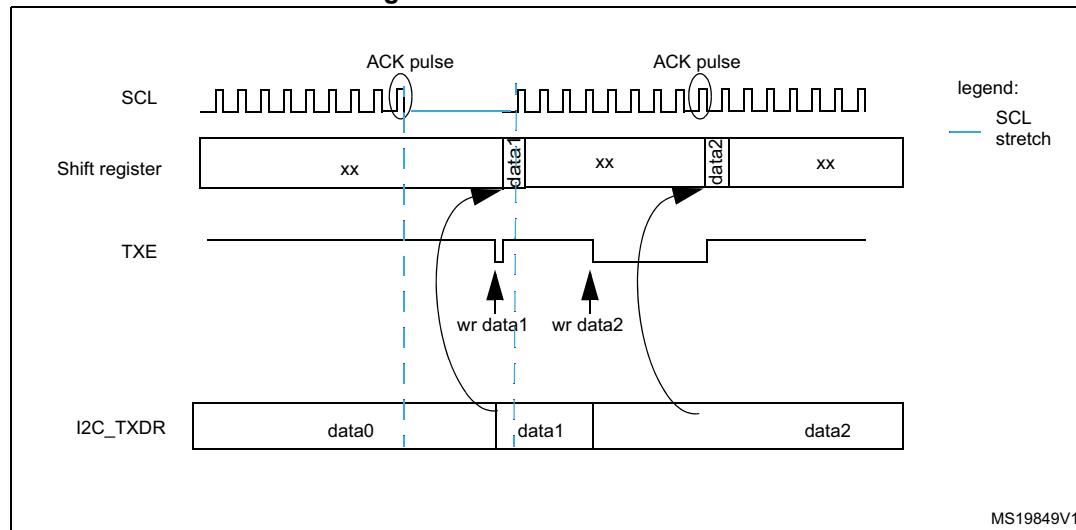
**Figure 215. Data reception**



## Transmission

If the I2C\_TXDR register is not empty ( $\text{TXE} = 0$ ), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on SDA line. If  $\text{TXE} = 1$ , meaning that no data is written yet in I2C\_TXDR, SCL line is stretched low until I2C\_TXDR is written. The stretch is done after the ninth SCL pulse.

**Figure 216. 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 I2C\_CR1 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C\_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES is 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 two modes:

- **Automatic end mode** (AUTOEND = '1' in the I2C\_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bit field is transferred.
- **Software end mode** (AUTOEND = '0' in the I2C\_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bit field is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit is set in the I2C\_CR2 register. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

**Table 92. I2C configuration**

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

### 23.4.8 I2C slave mode

#### I2C slave initialization

To work in slave mode, the user must enable at least one slave address. Registers I2C\_OAR1 and I2C\_OAR2 are available to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C\_OAR1 register.  
OA1 is enabled by setting the OA1EN bit in the I2C\_OAR1 register.
- If additional slave addresses are required, the second slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C\_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C\_OAR1 or I2C\_OAR2 register with OA2MSK = 0.

OA2 is enabled by setting the OA2EN bit in the I2C\_OAR2 register.

- The general call address is enabled by setting the GCEN bit in the I2C\_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, to perform software actions. If the master does not

support clock stretching, the I2C must be configured with NOSTRETCH = 1 in the I2C\_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C\_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

### Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C\_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE = 1). This stretch is released when the data is written to the I2C\_TXDR register.
- In reception when the I2C\_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C\_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC = 1 and RELOAD = 1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the 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 I2C\_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C\_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C\_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C\_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C\_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Slave byte control mode

In order to allow byte ACK control in slave reception mode, The Slave byte control mode must be enabled by setting the SBC bit in the I2C\_CR1 register. This is required to be compliant with SMBus standards.

The 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 eighth and ninth SCL pulses. The user can read the data from the I2C\_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C\_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

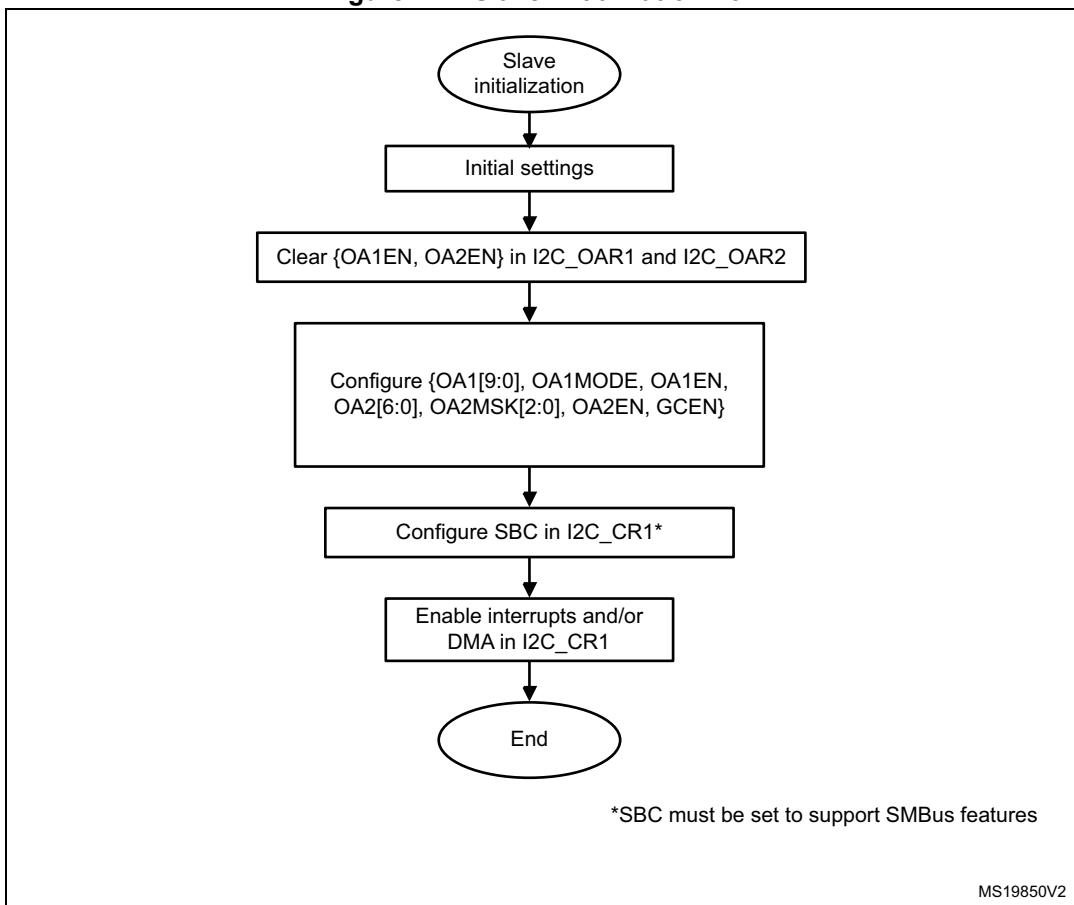
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

**Note:** *The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR = 1.*

*The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.*

**Caution:** The Slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

**Figure 217. Slave initialization flow**



### Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C\_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C\_CR1 register.

The TXIS bit is cleared when the I2C\_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C\_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C\_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C\_CR1 register, the STOPF flag is set in the I2C\_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to '0'. In this case, If TXE = 0 when the slave address is received (ADDR = 1), the user can choose either to send the content of the I2C\_TXDR register as the first data byte, or to flush the I2C\_TXDR register by setting the TXE bit in order to program a new data byte.

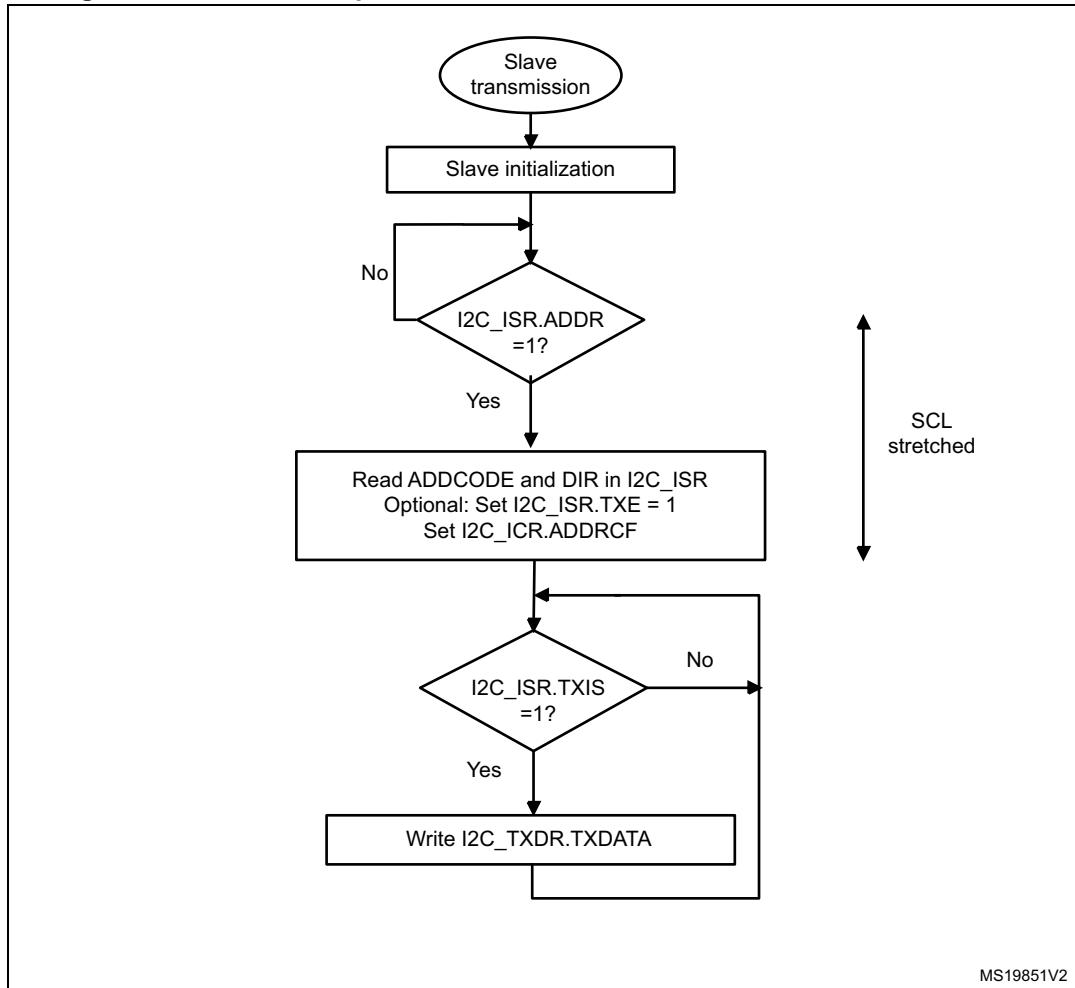
In Slave byte control mode (SBC = 1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

**Caution:** When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C\_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C\_TXDR register:

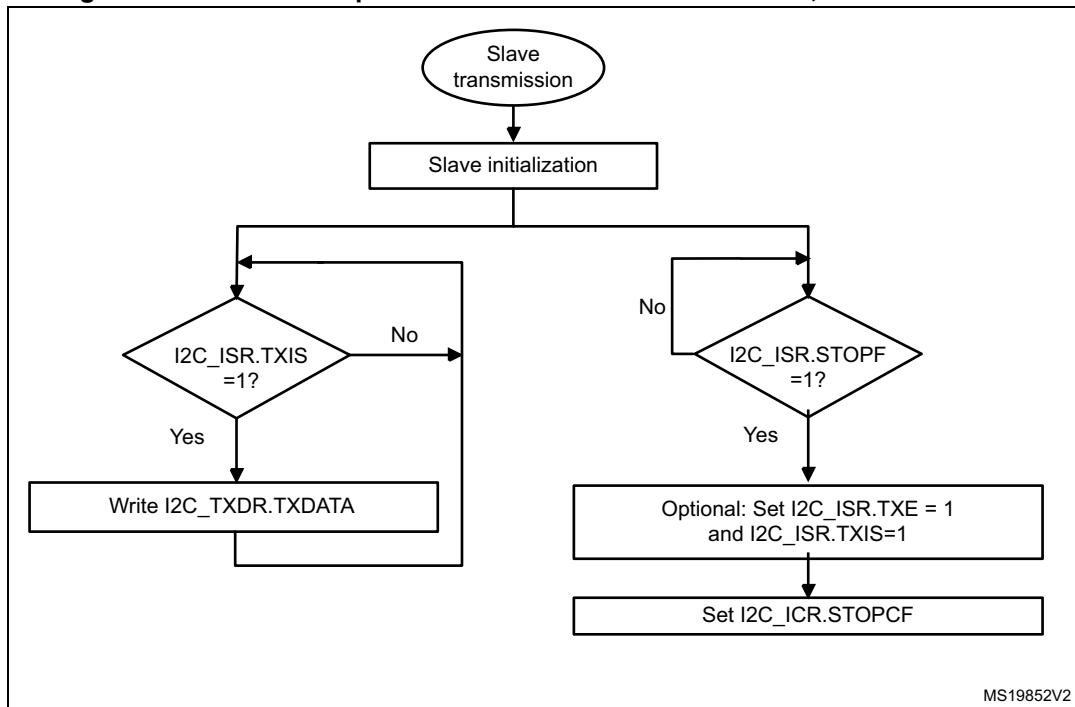
- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C\_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

If a TXIS event is needed, (transmit interrupt or transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

**Figure 218. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0**

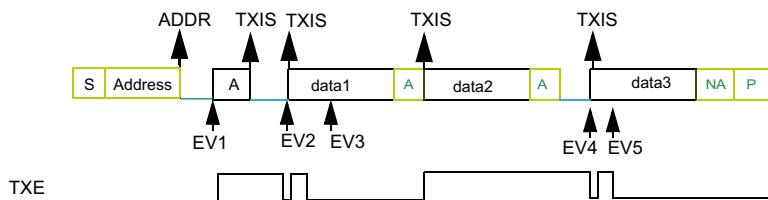
MS19851V2

**Figure 219. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1**

MS19852V2

**Figure 220. Transfer bus diagrams for I2C slave transmitter (mandatory events only)**

Example I2C slave transmitter 3 bytes with 1st data flushed,  
NOSTRETCH=0:



legend:

- transmission
- reception
- SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF

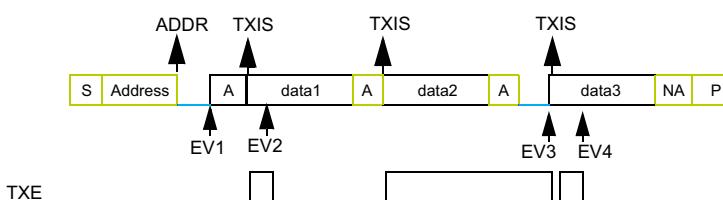
EV2: TXIS ISR: wr data1

EV3: TXIS ISR: wr data2

EV4: TXIS ISR: wr data3

EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush,  
NOSTRETCH=0:



legend :

- transmission
- reception
- SCL stretch

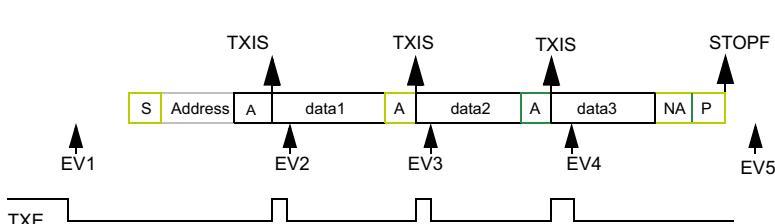
EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF

EV2: TXIS ISR: wr data2

EV3: TXIS ISR: wr data3

EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:



legend:

- transmission
- reception
- SCL stretch

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

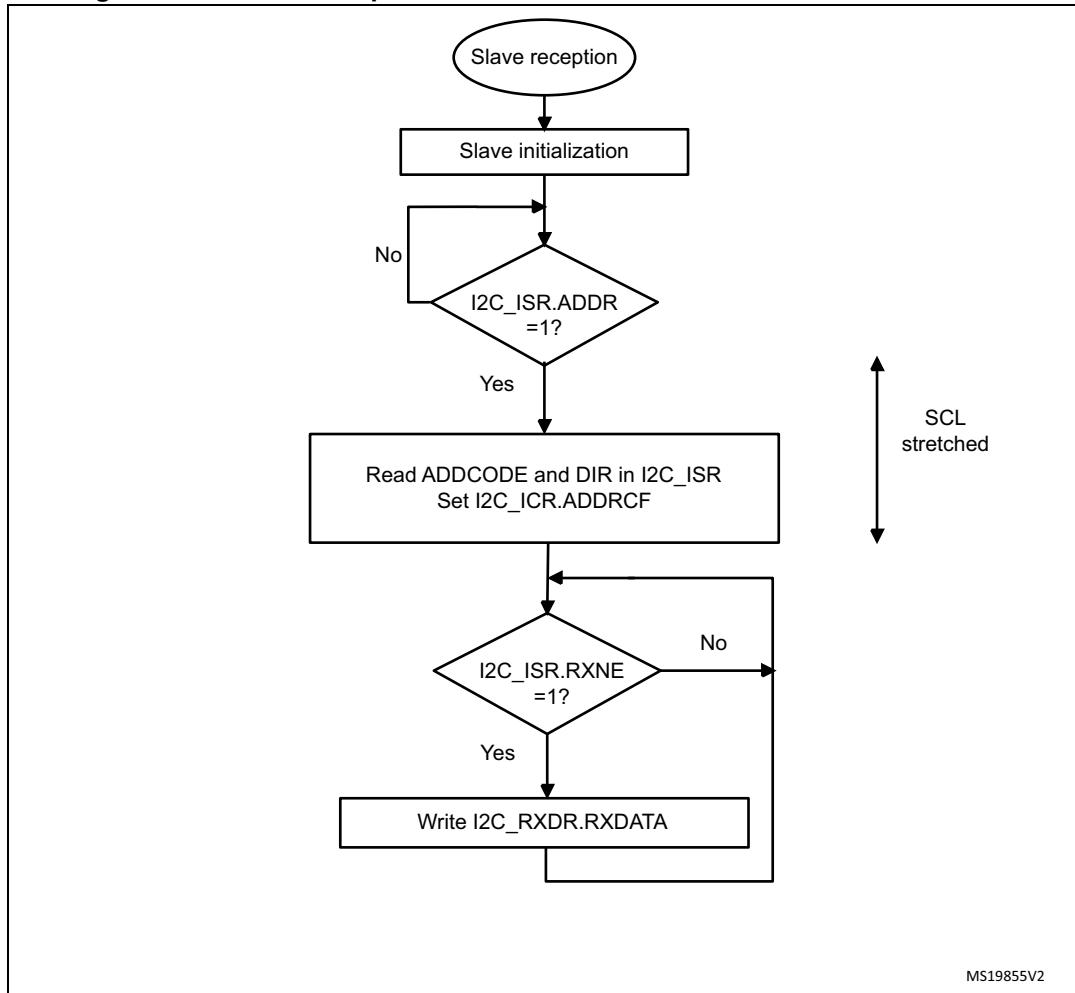
MS19853V2

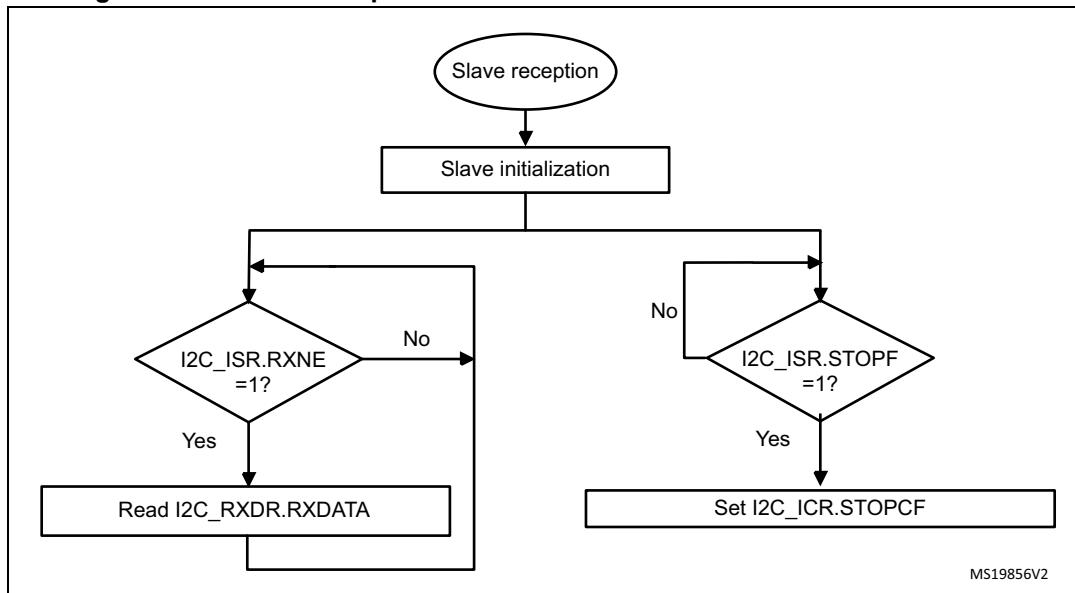
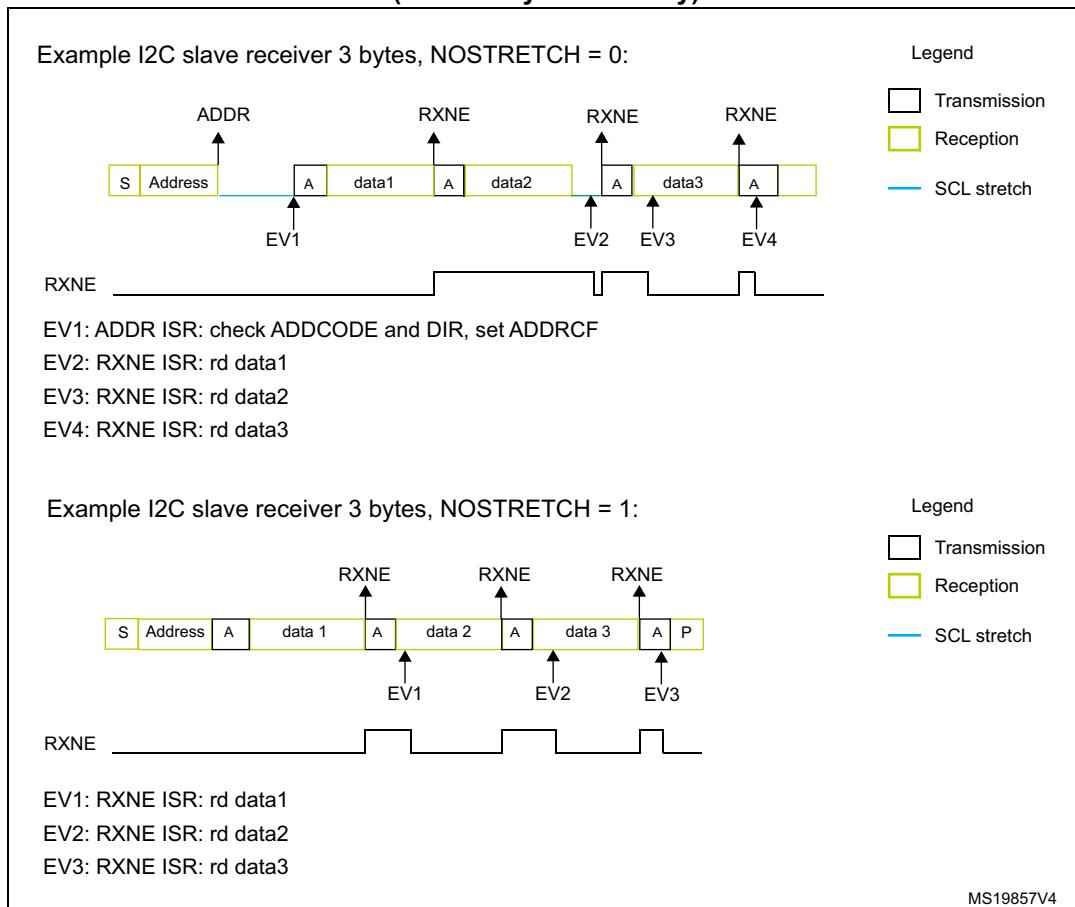
### Slave receiver

RXNE is set in I2C\_ISR when the I2C\_RXDR is full, and generates an interrupt if RXIE is set in I2C\_CR1. RXNE is cleared when I2C\_RXDR is read.

When a STOP is received and STOPIE is set in I2C\_CR1, STOPF is set in I2C\_ISR and an interrupt is generated.

**Figure 221. Transfer sequence flow for slave receiver with NOSTRETCH = 0**



**Figure 222. Transfer sequence flow for slave receiver with NOSTRETCH = 1****Figure 223. Transfer bus diagrams for I2C slave receiver (mandatory events only)**

### 23.4.9 I2C master mode

#### I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured by setting the SCLH and SCLL bits in the I2C\_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

The I2C detects its own SCL low level after a  $t_{SYNC1}$  delay depending on the SCL falling edge, SCL input noise filters (analog + digital) and SCL synchronization to the I2CxCLK clock. The I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bits in the I2C\_TIMINGR register.

The I2C detects its own SCL high level after a  $t_{SYNC2}$  delay depending on the SCL rising edge, SCL input noise filters (analog + digital) and SCL synchronization to I2CxCLK clock. The I2C ties SCL to low level once the SCLH counter is reached reaches the value programmed in the SCLH[7:0] bits in the I2C\_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$$

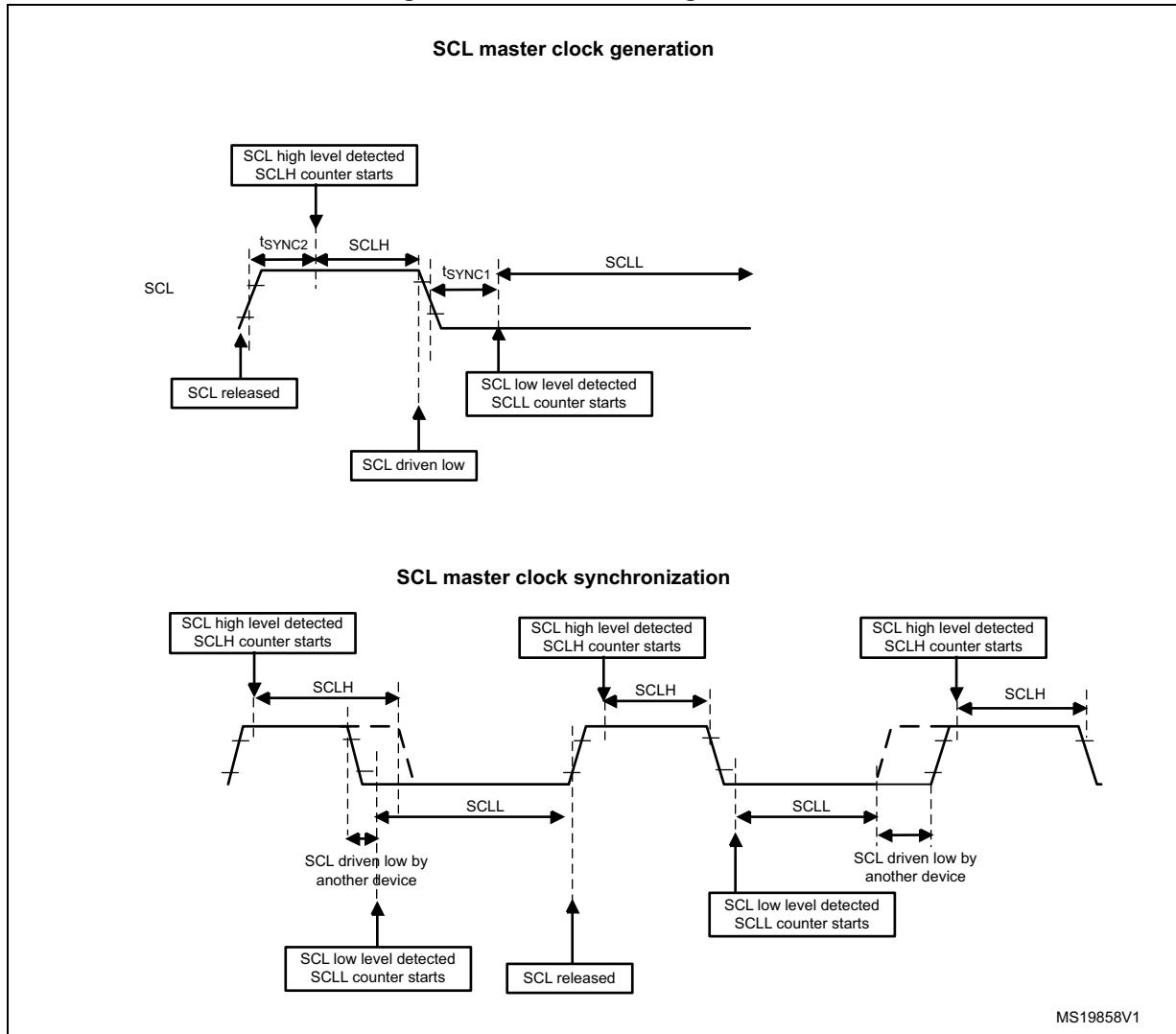
The duration of  $t_{SYNC1}$  depends on these parameters:

- SCL falling slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF  $\times t_{I2CCLK}$
- Delay due to SCL synchronization with I2CCLK clock (two to three I2CCLK periods)

The duration of  $t_{SYNC2}$  depends on these parameters:

- SCL rising slope
- When enabled, input delay induced by the analog filter.
- When enabled, input delay induced by the digital filter: DNF  $\times t_{I2CCLK}$
- Delay due to SCL synchronization with I2CCLK clock (two to three I2CCLK periods)

Figure 224. Master clock generation



**Caution:** To be I<sup>2</sup>C or SMBus compliant, the master clock must respect the timings given in the following table.

**Table 93. I<sup>2</sup>C-SMBus specification clock timings**

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		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	-	
t <sub>SU:STO</sub>	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t <sub>BUF</sub>	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t <sub>LOW</sub>	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t <sub>HIGH</sub>	Period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	ns
t <sub>r</sub>	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t <sub>f</sub>	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

**Note:** SCLL is also used to generate the t<sub>BUF</sub> and t<sub>SU:STA</sub> timings, and SCLH is also used to generate the t<sub>HD:STA</sub> and t<sub>SU:STO</sub> timings.

Refer to [Section 23.4.10](#) for examples of I2C\_TIMINGR settings vs. I2CCLK frequency.

### Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C\_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD\_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C\_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t<sub>BUF</sub>.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

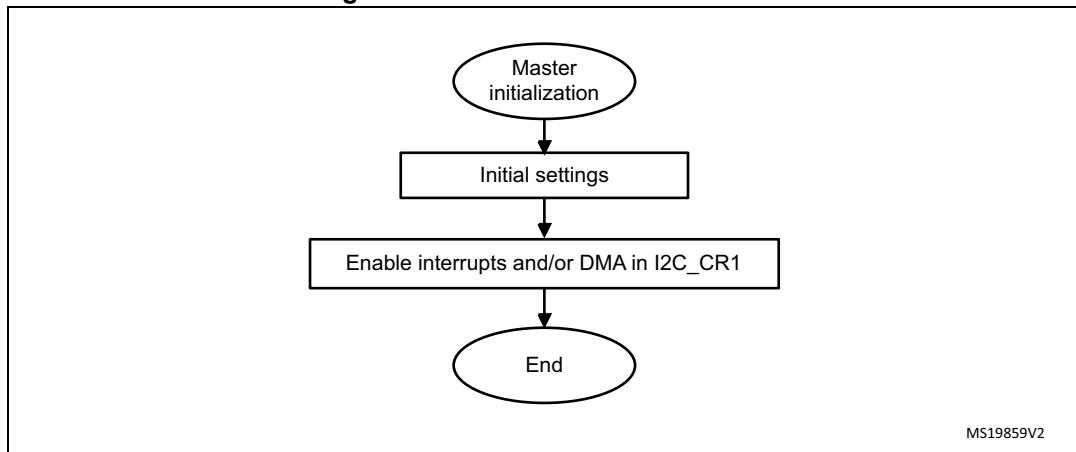
**Note:** The START bit is reset by hardware when the slave address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.

In 10-bit addressing mode, when the slave address first 7 bits are NACKed by the slave, the master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.

If the I2C is addressed as a slave (ADDR = 1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared.

**Note:** The same procedure is applied for a repeated start condition. In this case BUSY = 1.

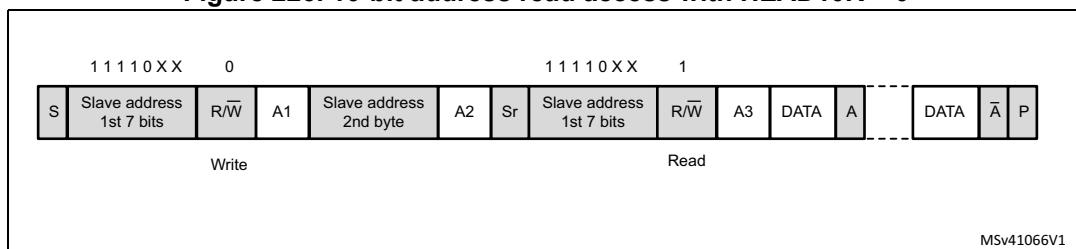
**Figure 225. Master initialization flow**



#### Initialization of a master receiver addressing a 10-bit address slave

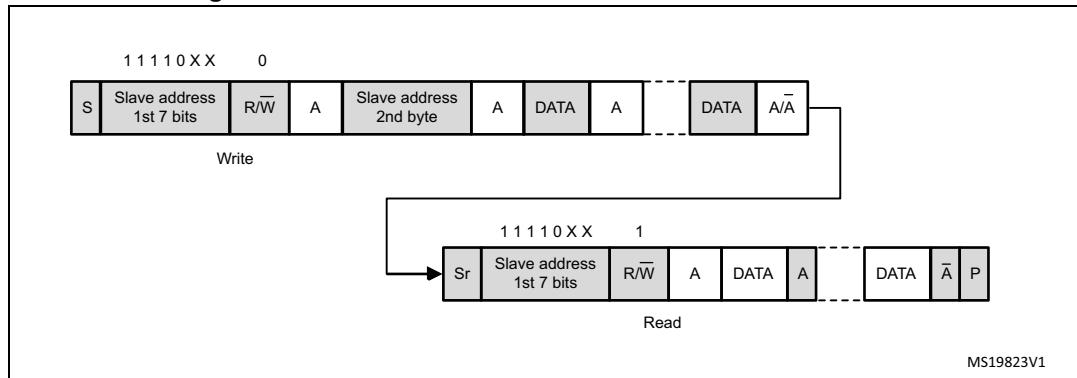
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C\_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address second byte + REStart + Slave address 10-bit header Read

**Figure 226. 10-bit address read access with HEAD10R = 0**



- If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated start is set with the 10 bit slave address configured with HEAD10R = 1. In this case the master sends this sequence: ReStart + Slave address 10-bit header Read.

**Figure 227. 10-bit address read access with HEAD10R = 1**



### Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C\_CR1 register. The flag is cleared when the I2C\_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD = 0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND = 1), a STOP is automatically sent.
  - In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low in order to perform software actions:
 

A RESTART condition can be requested by setting the START bit in the I2C\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2C\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C\_ISR register, and an interrupt is generated if the NACKIE bit is set.

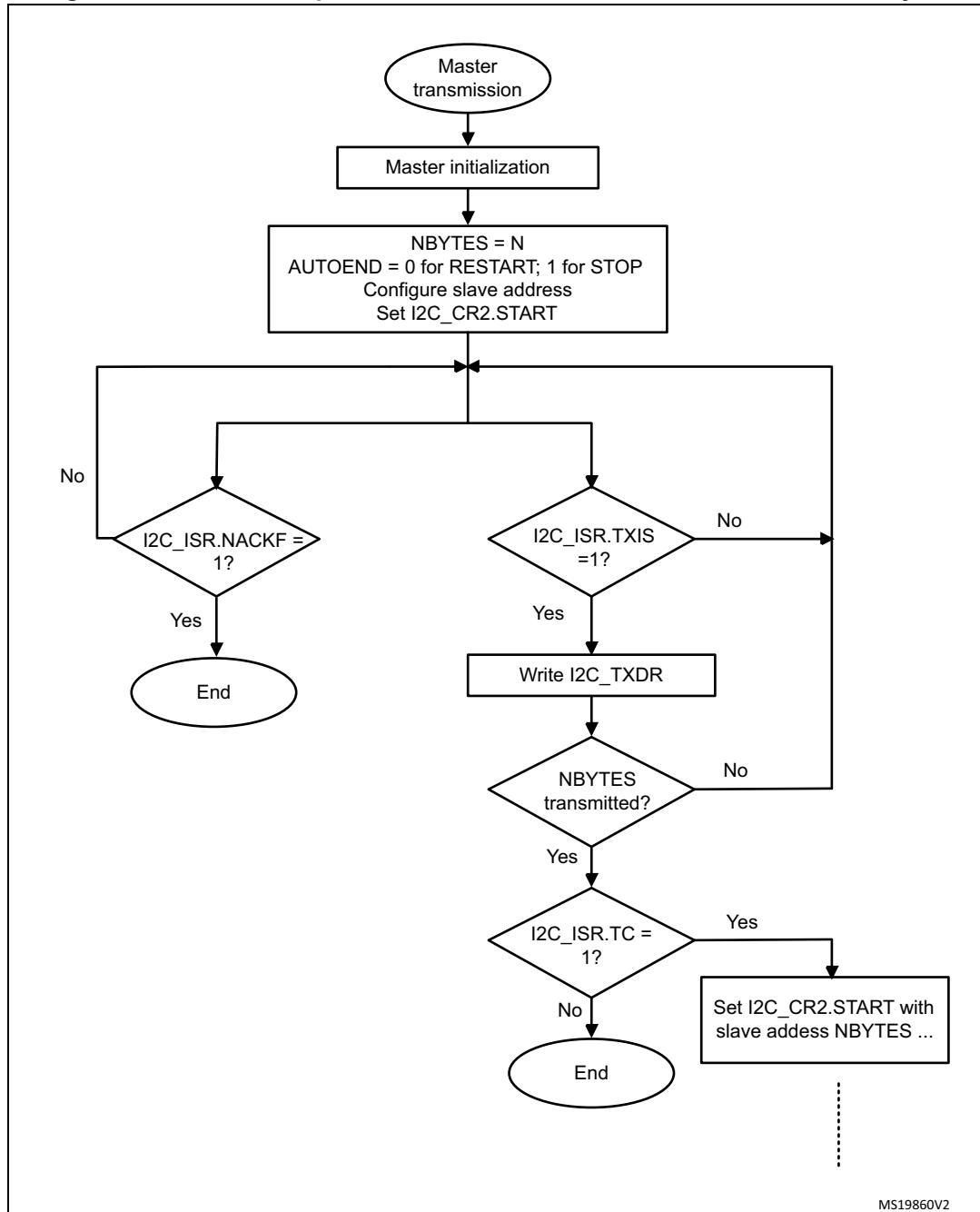
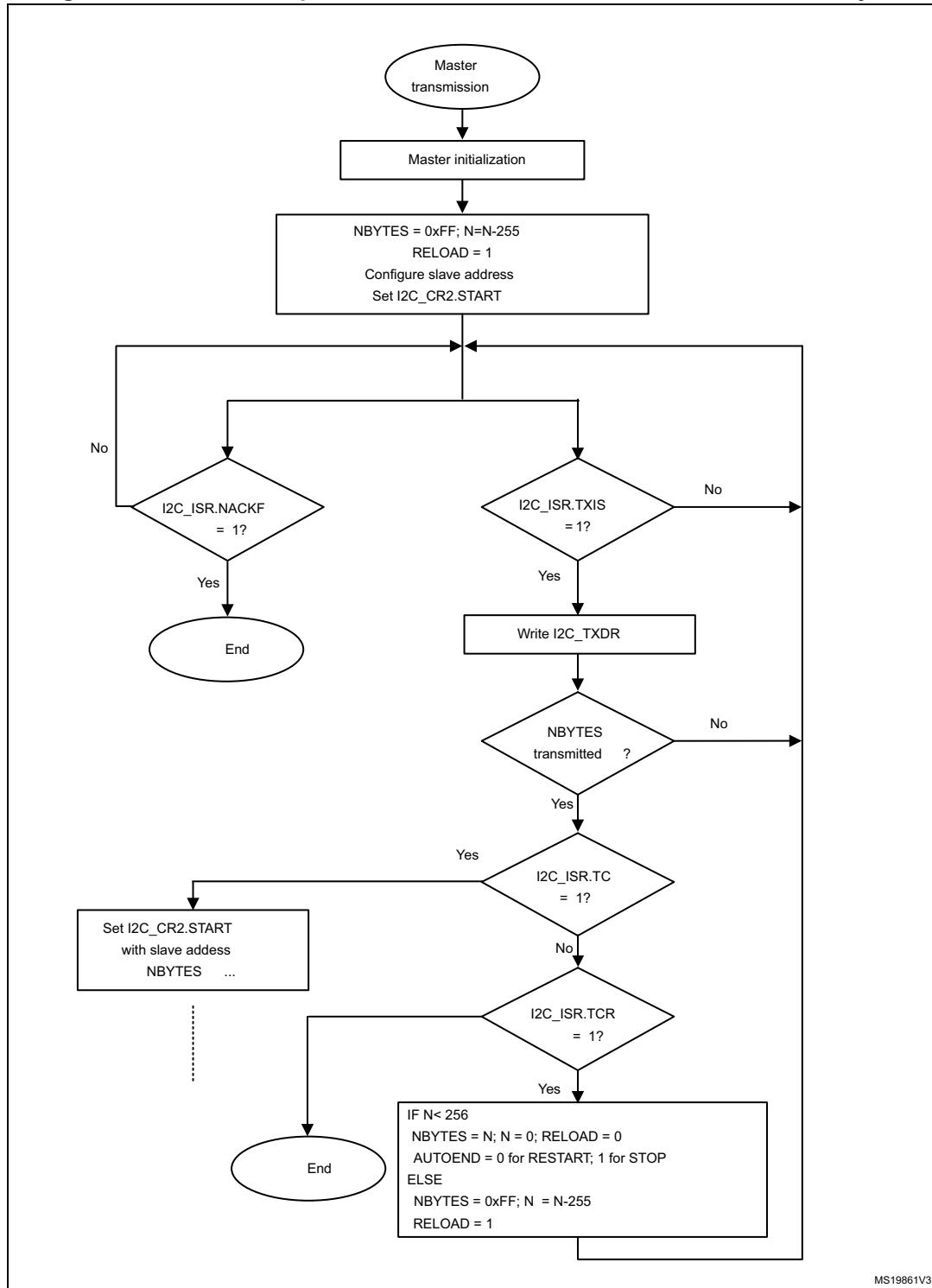
Figure 228. Transfer sequence flow for I2C master transmitter for  $N \leq 255$  bytes

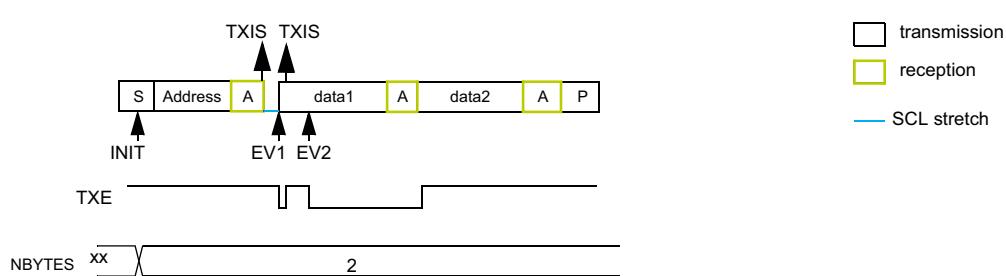
Figure 229. Transfer sequence flow for I2C master transmitter for N &gt; 255 bytes



MS19861V3

**Figure 230. Transfer bus diagrams for I2C master transmitter  
(mandatory events only)**

Example I2C master transmitter 2 bytes, automatic end mode (STOP)

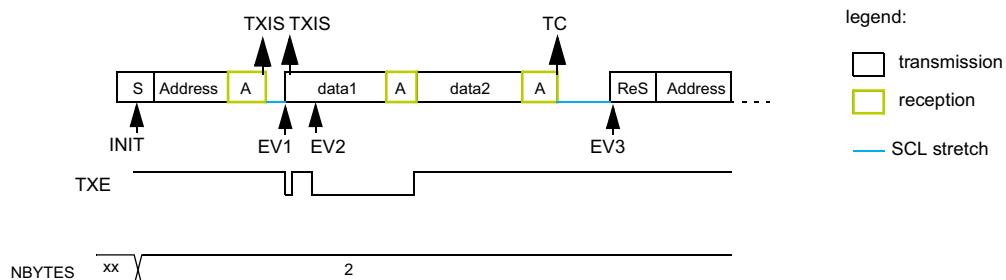


INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19862V2

### Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C\_CR1 register. The flag is cleared when I2C\_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD = 0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

A RESTART condition can be requested by setting the START bit in the I2C\_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

A STOP condition can be requested by setting the STOP bit in the I2C\_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

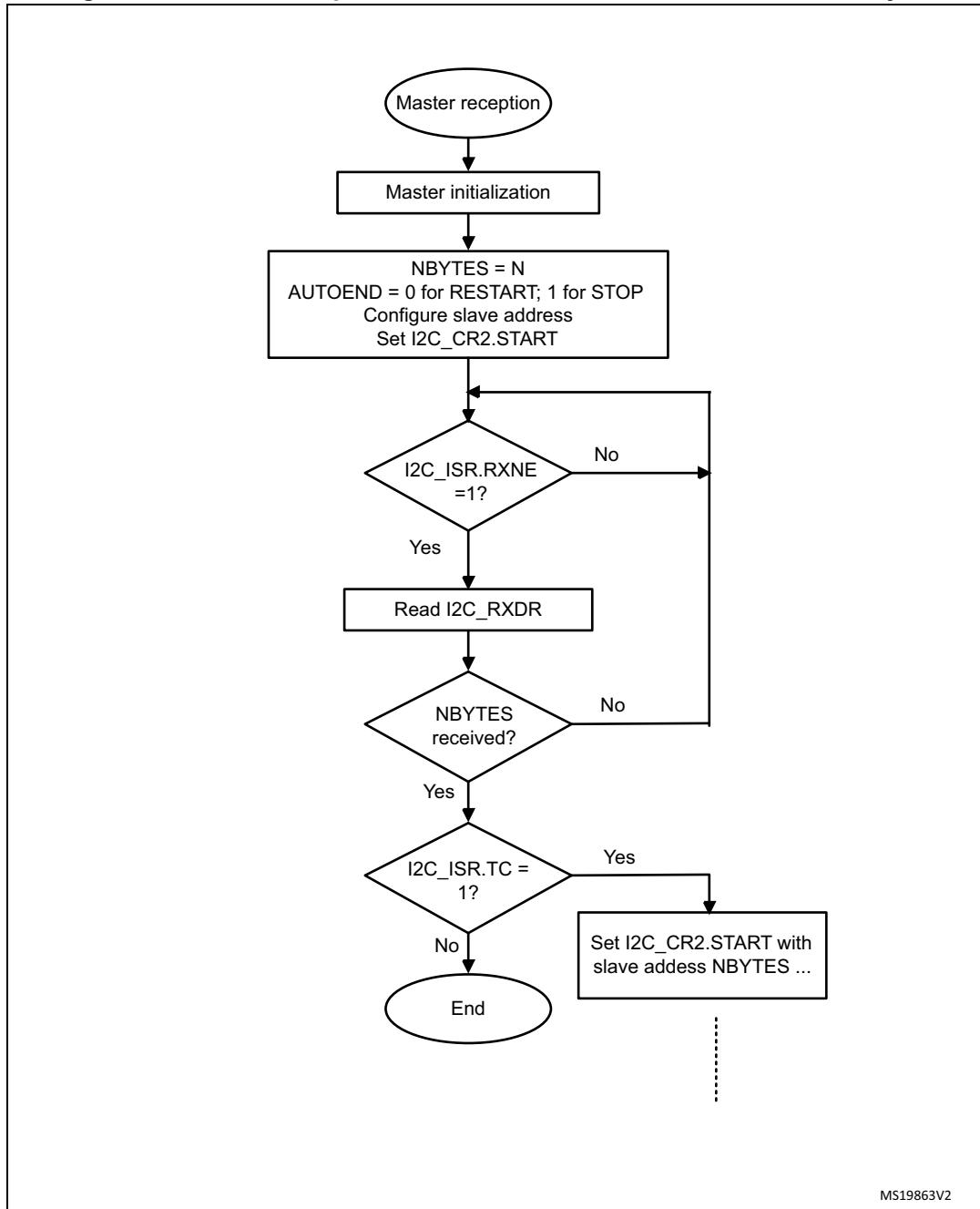
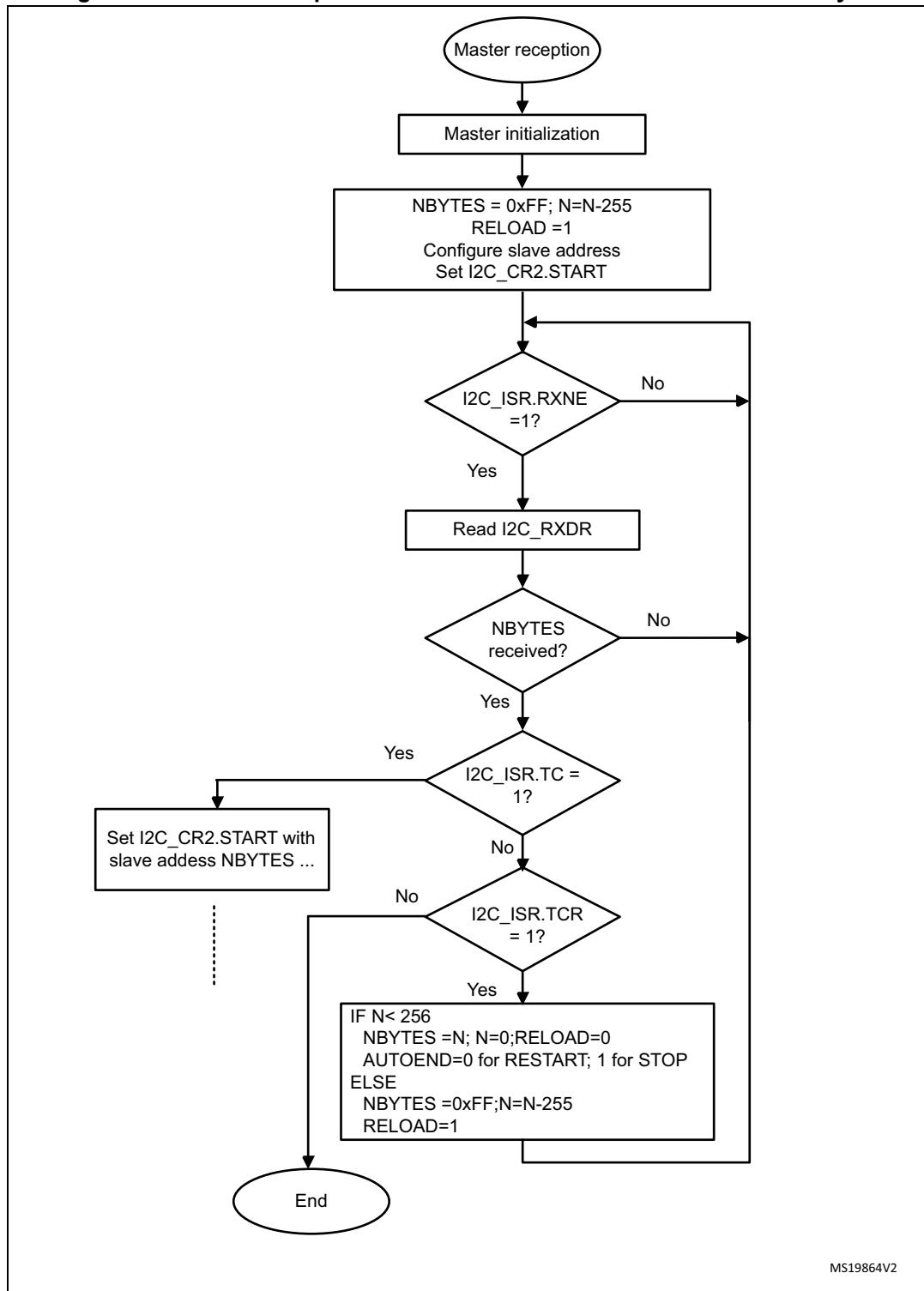
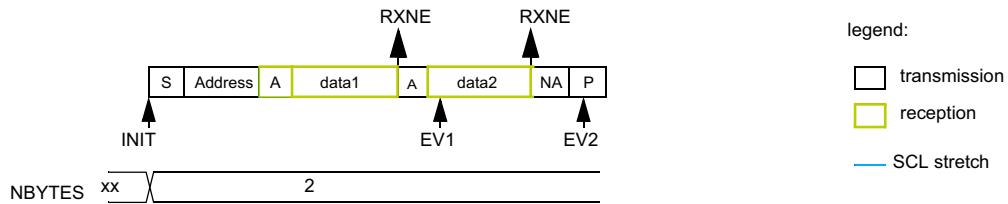
Figure 231. Transfer sequence flow for I2C master receiver for  $N \leq 255$  bytes

Figure 232. Transfer sequence flow for I2C master receiver for N &gt; 255 bytes



**Figure 233. Transfer bus diagrams for I2C master receiver  
(mandatory events only)**

Example I2C master receiver 2 bytes, automatic end mode (STOP)

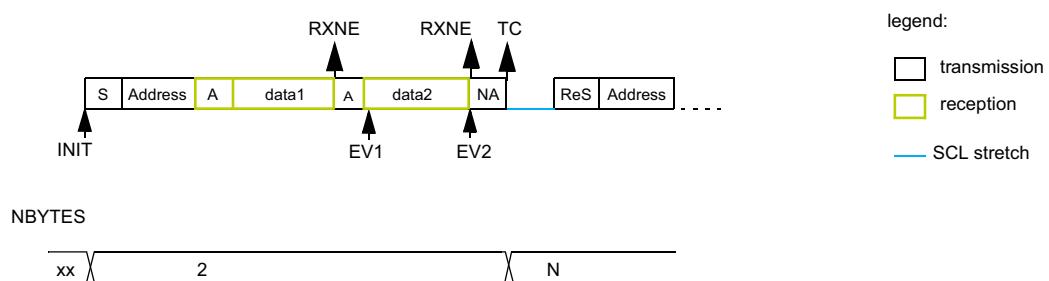


INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START

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

### 23.4.10 I2C\_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C\_TIMINGR to obtain timings compliant with the I<sup>2</sup>C specification. To get more accurate configuration values, use the STM32CubeMX tool (I2C Configuration window).

**Table 94. Examples of timing settings for  $f_{I2CCLK} = 8 \text{ MHz}$** 

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC	0x1	0x1	0x0	0x0
SCLL	0xC7	0x13	0x9	0x6
$t_{SCLL}$	$200 \times 250 \text{ ns} = 50 \mu\text{s}$	$20 \times 250 \text{ ns} = 5.0 \mu\text{s}$	$10 \times 125 \text{ ns} = 1250 \text{ ns}$	$7 \times 125 \text{ ns} = 875 \text{ ns}$
SCLH	0xC3	0xF	0x3	0x3
$t_{SCLH}$	$196 \times 250 \text{ ns} = 49 \mu\text{s}$	$16 \times 250 \text{ ns} = 4.0 \mu\text{s}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$
$t_{SCL}^{(1)}$	$\sim 100 \mu\text{s}^{(2)}$	$\sim 10 \mu\text{s}^{(2)}$	$\sim 2500 \text{ ns}^{(3)}$	$\sim 2000 \text{ ns}^{(4)}$
SDADEL	0x2	0x2	0x1	0x0
$t_{SDADEL}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$1 \times 125 \text{ ns} = 125 \text{ ns}$	0 ns
SCLDEL	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$2 \times 125 \text{ ns} = 250 \text{ ns}$

1.  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.

2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500 \text{ ns}$ . 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 95. Examples of timing settings for  $f_{I2CCLK} = 16 \text{ MHz}$** 

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0x3	0x3	0x1	0x0
SCLL	0xC7	0x13	0x9	0x4
$t_{SCLL}$	$200 \times 250 \text{ ns} = 50 \mu\text{s}$	$20 \times 250 \text{ ns} = 5.0 \mu\text{s}$	$10 \times 125 \text{ ns} = 1250 \text{ ns}$	$5 \times 62.5 \text{ ns} = 312.5 \text{ ns}$
SCLH	0xC3	0xF	0x3	0x2
$t_{SCLH}$	$196 \times 250 \text{ ns} = 49 \mu\text{s}$	$16 \times 250 \text{ ns} = 4.0 \mu\text{s}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$3 \times 62.5 \text{ ns} = 187.5 \text{ ns}$
$t_{SCL}^{(1)}$	$\sim 100 \mu\text{s}^{(2)}$	$\sim 10 \mu\text{s}^{(2)}$	$\sim 2500 \text{ ns}^{(3)}$	$\sim 1000 \text{ ns}^{(4)}$
SDADEL	0x2	0x2	0x2	0x0
$t_{SDADEL}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$2 \times 125 \text{ ns} = 250 \text{ ns}$	0 ns
SCLDEL	0x4	0x4	0x3	0x2
$t_{SCLDEL}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$3 \times 62.5 \text{ ns} = 187.5 \text{ ns}$

1.  $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.

2.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}$ .

3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}$ .

4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 500 \text{ ns}$ .

**Table 96. Examples of timing settings for  $f_{I2CCLK} = 48$  MHz**

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC	0xB	0xB	0x5	0x5
SCLL	0xC7	0x13	0x9	0x3
$t_{SCLL}$	$200 \times 250 \text{ ns} = 50 \mu\text{s}$	$20 \times 250 \text{ ns} = 5.0 \mu\text{s}$	$10 \times 125 \text{ ns} = 1250 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$
SCLH	0xC3	0xF	0x3	0x1
$t_{SCLH}$	$196 \times 250 \text{ ns} = 49 \mu\text{s}$	$16 \times 250 \text{ ns} = 4.0 \mu\text{s}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$2 \times 125 \text{ ns} = 250 \text{ ns}$
$t_{SCL}^{(1)}$	$\sim 100 \mu\text{s}^{(2)}$	$\sim 10 \mu\text{s}^{(2)}$	$\sim 2500 \text{ ns}^{(3)}$	$\sim 875 \text{ ns}^{(4)}$
SDADEL	0x2	0x2	0x3	0x0
$t_{SDADEL}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$2 \times 250 \text{ ns} = 500 \text{ ns}$	$3 \times 125 \text{ ns} = 375 \text{ ns}$	0 ns
SCLDEL	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$5 \times 250 \text{ ns} = 1250 \text{ ns}$	$4 \times 125 \text{ ns} = 500 \text{ ns}$	$2 \times 125 \text{ ns} = 250 \text{ ns}$

1.  $t_{SCL}$  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  $4 \times t_{I2CCLK} = 83.3 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 1000 \text{ ns}$

3.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 750 \text{ ns}$

4.  $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3 \text{ ns}$ . Example with  $t_{SYNC1} + t_{SYNC2} = 250 \text{ ns}$

### 23.4.11 SMBus specific features

This section is relevant only when the SMBus feature is supported (refer to [Section 23.3](#)).

#### 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. The SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (<http://smbus.org>).

The system management bus specification refers to three types of devices

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

#### Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block

Read, Block Write and Block Write-Block Read Process Call. These protocols must be implemented by the user software.

For more details on these protocols, refer to SMBus specification (<http://smbus.org>).

### Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C\_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus address resolution protocol, refer to SMBus specification (<http://smbus.org>).

### Received command and data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C\_CR1 register. Refer to *Slave byte control mode* for more details.

### Host notify protocol

This peripheral supports the host notify protocol by setting the SMBHEN bit in the I2C\_CR1 register. In this case the host acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

### SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the alert response address.

When configured as a slave device(SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C\_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN = 1), the ALERT flag is set in the I2C\_ISR register when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register. When ALERTEN = 0, the ALERT line is considered high even if the external SMBA pin is low.

*If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN = 0.*

### Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer.

The PEC is calculated by using the  $C(x) = x_8 + x^2 + x + 1$  CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

### Timeouts

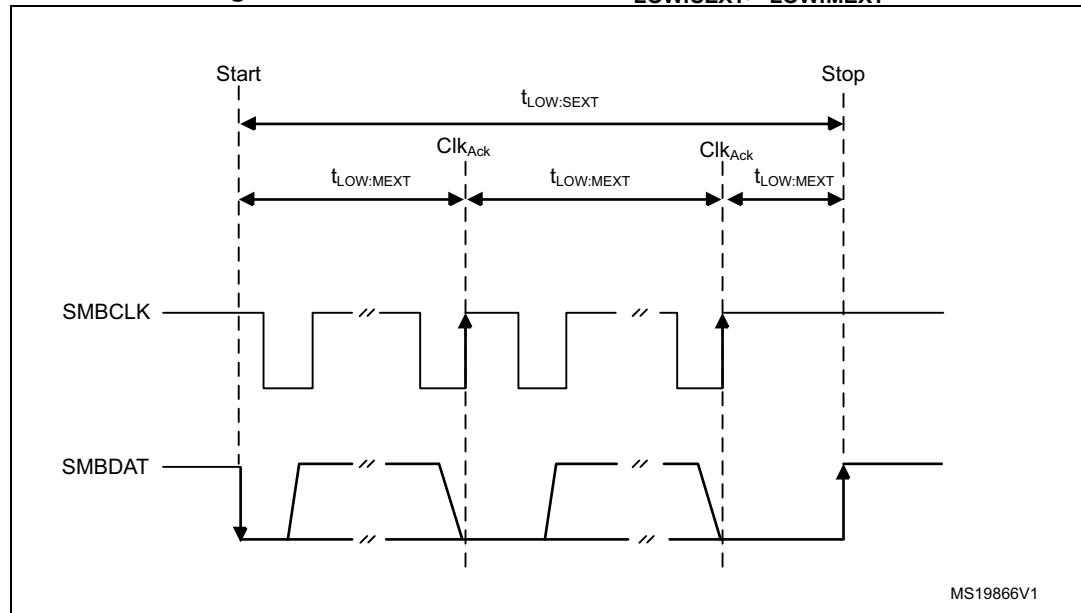
This peripheral embeds hardware timers in order to be compliant with the three timeouts defined in SMBus specification.

**Table 97. SMBus timeout specifications**

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}^{(1)}$	Cumulative clock low extend time (slave device)	-	25	
$t_{LOW:MEXT}^{(2)}$	Cumulative clock low extend time (master device)	-	10	

1.  $t_{LOW:SEXT}$  is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that, another slave device or the master also extends the clock causing the combined clock low extend time to be greater than  $t_{LOW:SEXT}$ . Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2.  $t_{LOW:MEXT}$  is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock, causing the combined clock low time to be greater than  $t_{LOW:MEXT}$  on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

**Figure 234. 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 91](#)).

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.

### 23.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported (see [Section 23.3](#)).

In addition to I2C initialization, some other specific initialization must be done to perform SMBus communication.

#### Received command and data acknowledge control (slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave byte control mode must be enabled by setting the SBC bit in the I2C\_CR1 register. Refer to [Slave byte control mode](#) for more details.

#### Specific address (slave mode)

The specific SMBus addresses must be enabled if needed. Refer to [Bus idle detection](#) for more details.

- The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit in the I2C\_CR1 register.
- The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit in the I2C\_CR1 register.
- The alert response address (0b0001100) is enabled by setting the ALERTEN bit in the I2C\_CR1 register.

#### Packet error checking

PEC calculation is enabled by setting the PECEN bit in the I2C\_CR1 register. Then the PEC transfer is managed with the help of a hardware byte counter: NBYTES[7:0] in the I2C\_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after NBYTES - 1 data have been transferred when the PECPBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECPBYTE has no effect.

**Caution:** Changing the PECEN configuration is not allowed when the I2C is enabled.

**Table 98. SMBus with PEC configuration**

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECPBYTE 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 I2C\_TIMEOUTTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

- $t_{TIMEOUT}$  check

To enable the  $t_{TIMEOUT}$  check, the 12-bit TIMEOUTA[11:0] bits must be programmed with the timer reload value, to check the  $t_{TIMEOUT}$  parameter. The TIDLE bit must be configured to '0' to detect the SCL low level timeout.

Then the timer is enabled by setting the TIMOUTEN in the I2C\_TIMEOUTTR 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 I2C\_ISR register.

Refer to [Table 99](#).

**Caution:** Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMOUTEN 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, the user can choose the same value for the both.

Then the timer is enabled by setting the TEXTEN bit in the I2C\_TIMEOUTTR 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](#) section, the TIMEOUT flag is set in the I2C\_ISR register.

Refer to [Table 100](#)

**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 I2C\_TIMEOUTTR 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 I2C\_ISR register.

Refer to [Table 101](#).

**Caution:** Changing TIMEOUTA and TIDLE configuration is not allowed when TIMOUTEN is set.

### 23.4.13 SMBus: I2C\_TIMEOUTTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to [Section 23.3](#).

- Configuring the maximum duration of  $t_{TIMEOUT}$  to 25 ms:

**Table 99. Examples of TIMEOUTA settings (max  $t_{TIMEOUT} = 25$  ms)**

$f_{I2CCLK}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	$98 \times 2048 \times 125$ ns = 25 ms
16 MHz	0xC3	0	1	$196 \times 2048 \times 62.5$ ns = 25 ms
48 MHz	0x249	0	1	$586 \times 2048 \times 20.08$ ns = 25 ms

- Configuring the maximum duration of  $t_{LOW:SEXT}$  and  $t_{LOW:MEXT}$  to 8 ms:

**Table 100. Examples of TIMEOUTB settings**

$f_{I2CCLK}$	TIMEOUTB[11:0] bits	TEXten bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	$32 \times 2048 \times 125$ ns = 8 ms
16 MHz	0x3F	1	$64 \times 2048 \times 62.5$ ns = 8 ms
48 MHz	0xBB	1	$188 \times 2048 \times 20.08$ ns = 8 ms

- Configuring the maximum duration of  $t_{IDLE}$  to 50  $\mu$ s

**Table 101. Examples of TIMEOUTA settings (max  $t_{IDLE} = 50$   $\mu$ s)**

$f_{I2CCLK}$	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIDLE}$
8 MHz	0x63	1	1	$100 \times 4 \times 125$ ns = 50 $\mu$ s
16 MHz	0xC7	1	1	$200 \times 4 \times 62.5$ ns = 50 $\mu$ s
48 MHz	0x257	1	1	$600 \times 4 \times 20.08$ ns = 50 $\mu$ s

### 23.4.14 SMBus slave mode

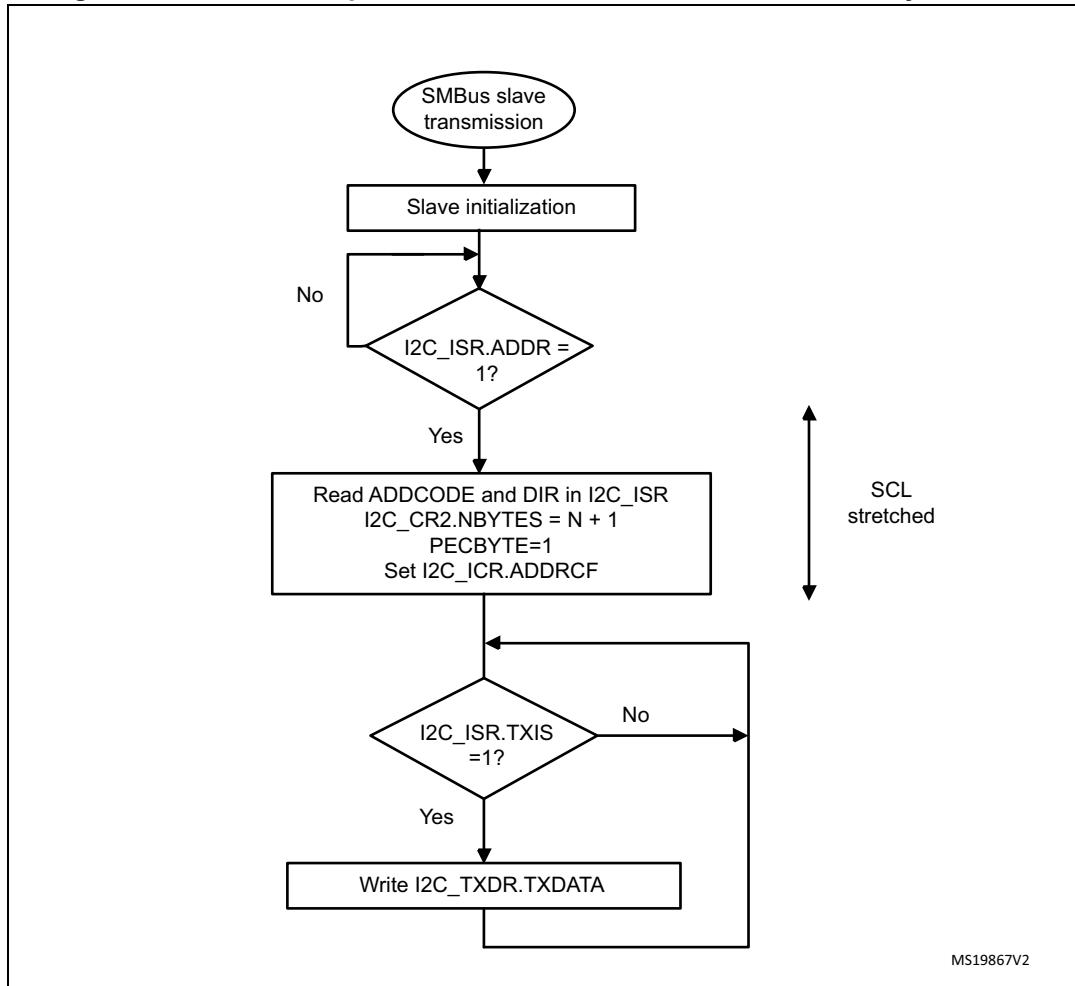
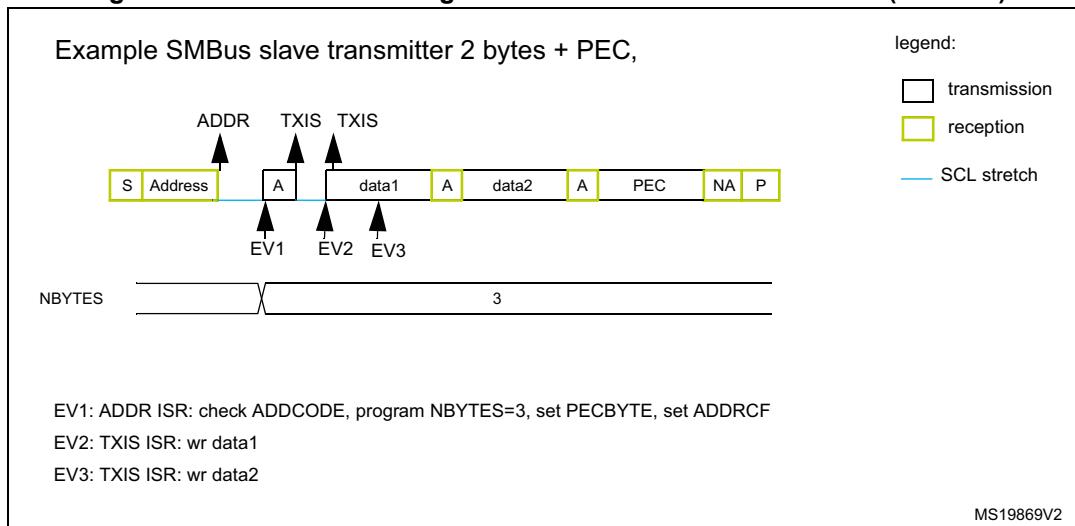
This section is relevant only when the SMBus feature is supported (refer to [Section 23.3](#)).

In addition to I2C slave transfer management (refer to [Section 23.4.8](#)), some additional software flows are provided to support the SMBus.

#### SMBus slave transmitter

When the IP is used in SMBus, SBC must be programmed to '1' to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES - 1 and the content of the I2C\_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES - 1 data transfer.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 235. Transfer sequence flow for SMBus slave transmitter N bytes + PEC****Figure 236. 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' 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](#) for more details.

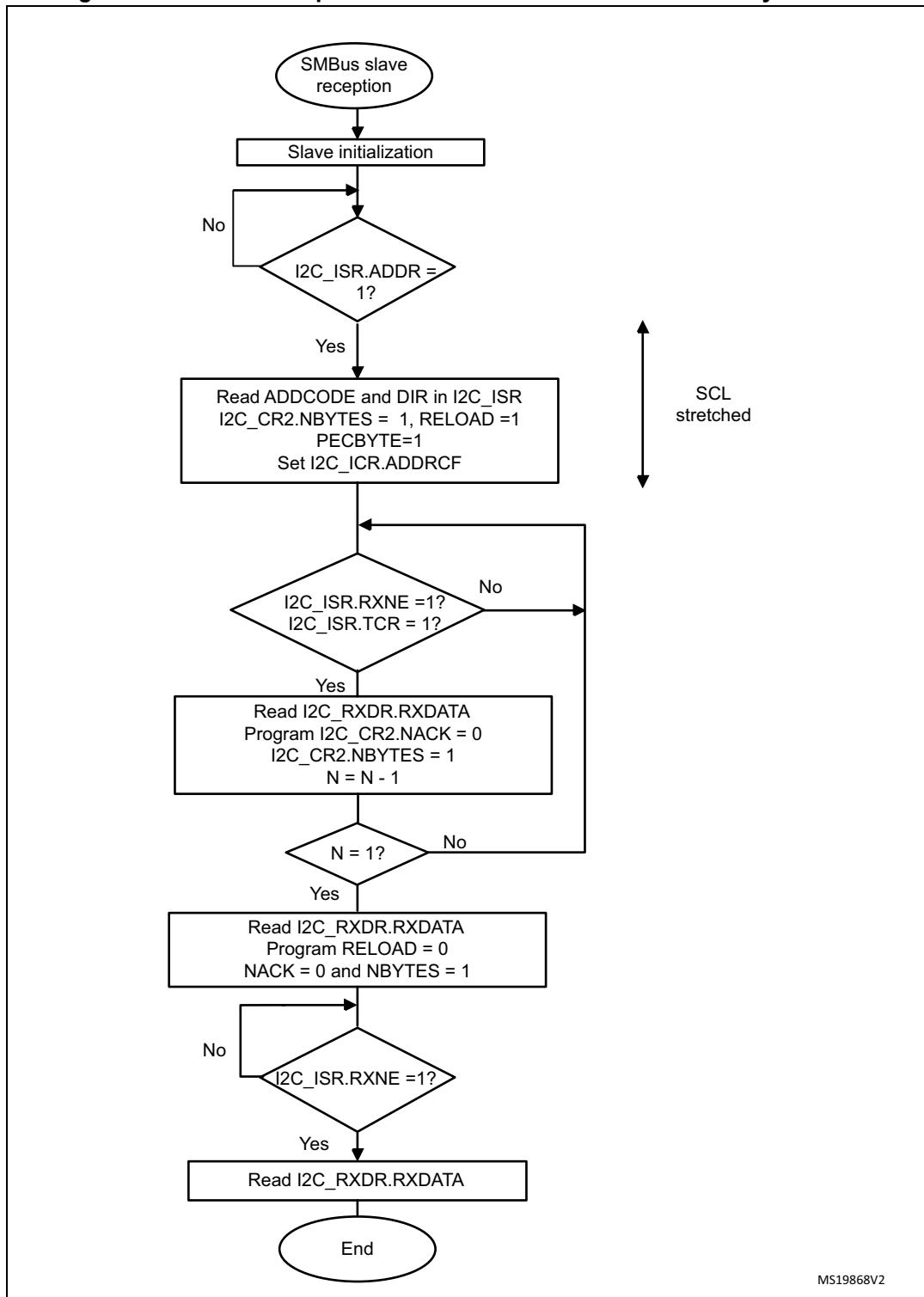
In order to check the PEC byte, the RELOAD bit must be cleared and the PECPBYTE bit must be set. In this case, after NBYTES - 1 data have been received, the next received byte is compared with the internal I2C\_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C\_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECPERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

If no ACK software control is needed, the user can program PECPBYTE = 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.

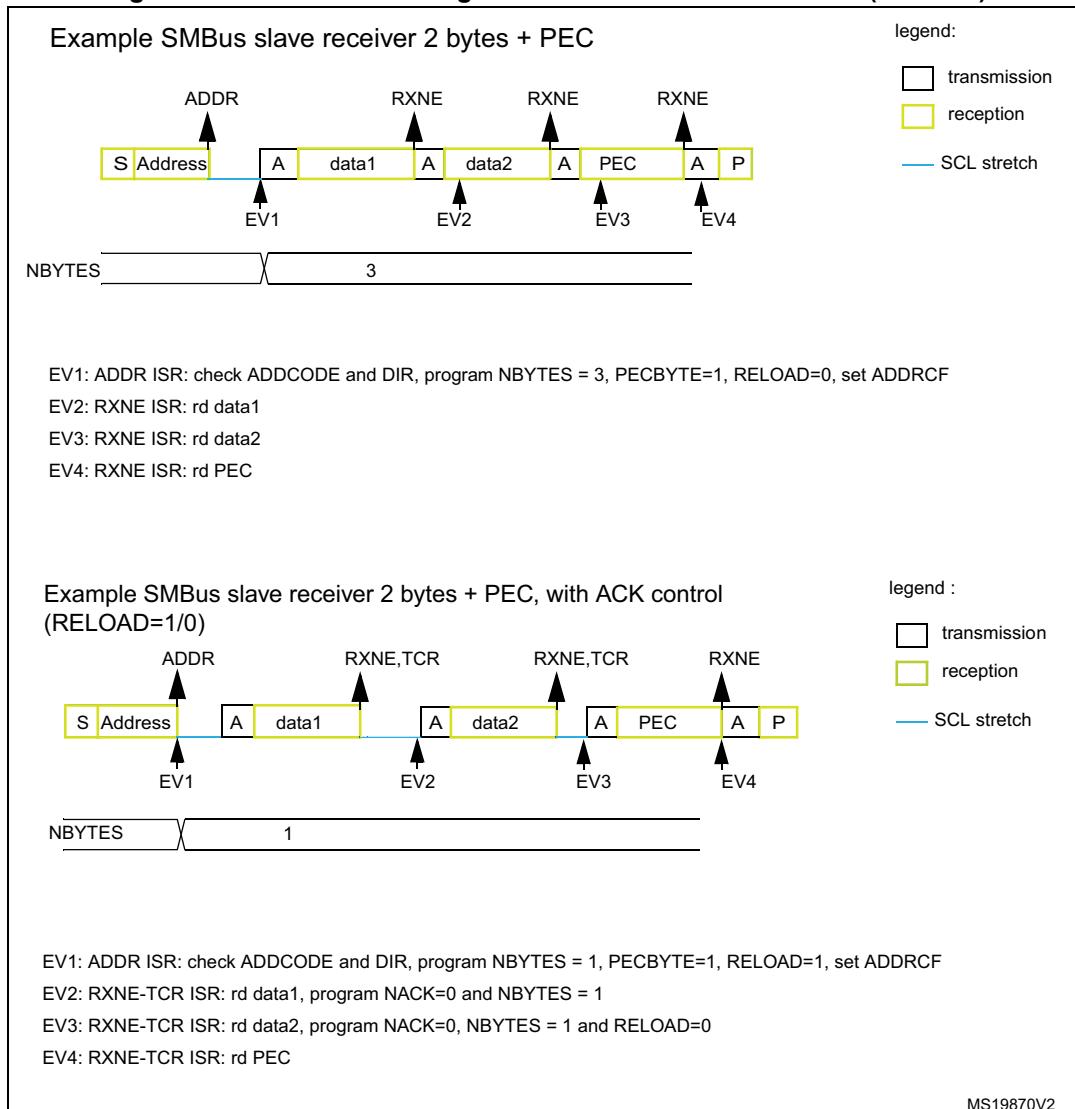
**Caution:** The PECPBYTE bit has no effect when the RELOAD bit is set.

Figure 237. Transfer sequence flow for SMBus slave receiver N bytes + PEC



MS19868V2

Figure 238. Bus transfer diagrams for SMBus slave receiver (SBC = 1)



This section is relevant only when the SMBus feature is supported (refer to [Section 23.3](#)).

In addition to I2C master transfer management (refer to [Section 23.4.9](#)), some additional software flows are provided to support the SMBus.

### SMBus master transmitter

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES - 1. So if the PECBYTE bit is set when NBYTES = 0x1, the content of the I2C\_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

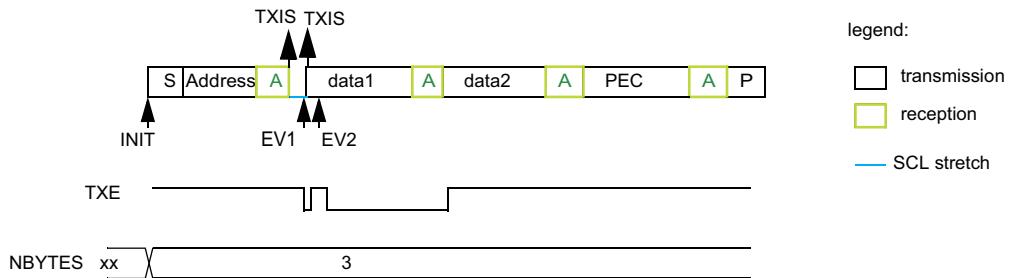
When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND = 0). In this case, once NBYTES - 1 have been

transmitted, the I2C\_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

**Caution:** The PECPBYTE bit has no effect when the RELOAD bit is set.

Figure 239. Bus transfer diagrams for SMBus master transmitter

Example SMBus master transmitter 2 bytes + PEC, automatic end mode (STOP)

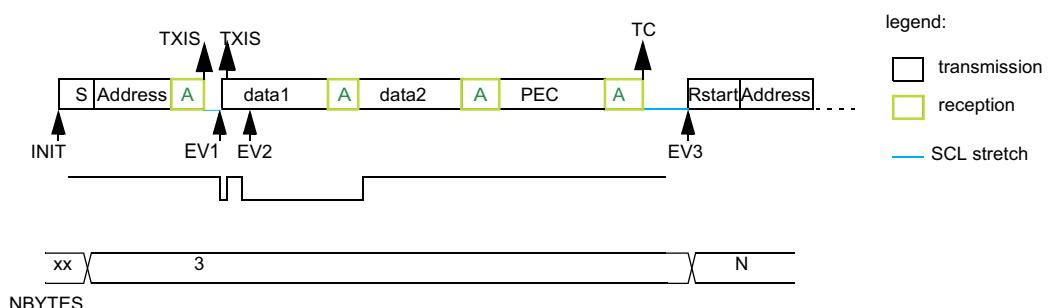


INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECPBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

Example SMBus master transmitter 2 bytes + PEC, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECPBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

EV3: TC ISR: program Slave address, program NBYTES = N, set START

MS19871V2

### SMBus master receiver

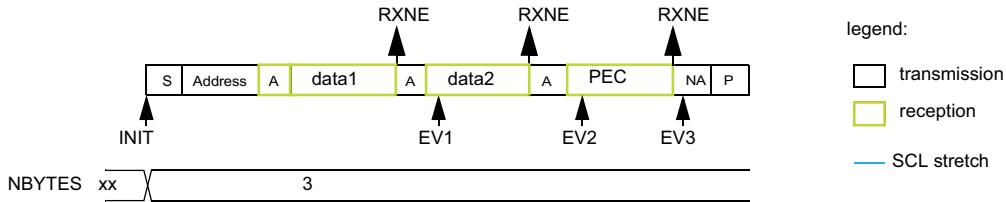
When the SMBus master wants to receive the PEC followed by a STOP at the end of the transfer, automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C\_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive the PEC byte followed by a RESTART condition at the end of the transfer, software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the slave address must be programmed, before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C\_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 240. Bus transfer diagrams for SMBus master receiver**

Example SMBus master receiver 2 bytes + PEC, automatic end mode (STOP)



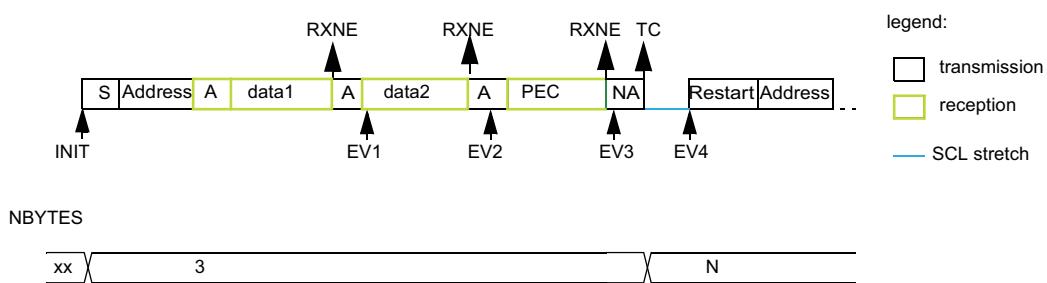
INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECPBYTE, set START

EV1: RXNE ISR: rd data1

EV2: RXNE ISR: rd data2

EV3: RXNE ISR: rd PEC

Example SMBus master receiver 2 bytes + PEC, software end mode (RESTART)



INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECPBYTE, set START

EV1: RXNE ISR: rd data1

EV2: RXNE ISR: rd data2

EV3: RXNE ISR: read PEC

EV4: TC ISR: program Slave address, program NBYTES = N, set START

MS19872V2

### 23.4.15 Wake-up from Stop mode on address match

This section is relevant only when wake-up from Stop mode feature is supported (refer to [Section 23.3](#)).

The I2C is able to wake-up the MCU from Stop mode (APB clock is off), when it is addressed. All addressing modes are supported.

Wake-up from Stop mode is enabled by setting the WUPEN bit in the I2C\_CR1 register. The HSI48 oscillator must be selected as the clock source for I2CCLK in order to allow wake-up from Stop mode.

During Stop mode, the HSI48 is switched off. When a START is detected, the I2C interface switches the HSI48 on, and stretches SCL low until HSI48 is woken up.

HSI48 is then used for the address reception.

In case of an address match, the I2C stretches SCL low during MCU wake-up 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 HSI48 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 HSI48 is not switched on after a START is received.*

*Only an ADDR interrupt can wake-up 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 wake-up 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 HSI48 oscillator.

**Caution:** Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

**Caution:** If wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

### 23.4.16 Error conditions

The following errors are the conditions that can cause a communication fail.

#### Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. A START or a STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C is involved in the transfer as master or addressed slave (i.e not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

#### Arbitration lost (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

- In master mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the master switches automatically to slave mode.
- In slave mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped, and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Overrun/underrun error (OVR)

An overrun or underrun error is detected in slave mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet.  
The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C\_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C\_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Packet error checking error (PECERR)

This section is relevant only when the SMBus feature is supported (refer to [Section 23.3](#)).

A PEC error is detected when the received PEC byte does not match with the I2C\_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Timeout error (TIMEOUT)

This section is relevant only when the SMBus feature is supported (refer to [Section 23.3](#)).

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 an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remained high for the time defined in the TIMEOUTA [11:0] bits: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reached the time defined in the TIMEOUTB[11:0] bits (SMBus  $t_{LOW:MEXT}$  parameter).
- Slave cumulative clock low extend time reached the time defined in TIMEOUTB[11:0] bits (SMBus  $t_{LOW:SEXT}$  parameter).

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### Alert (ALERT)

This section is relevant only when the SMBus feature is supported (refer to [Section 23.3](#)).

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN = 1), the alert pin detection is enabled (ALERTEN = 1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C\_CR1 register.

### 23.4.17 DMA requests

#### Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit in the I2C\_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see [Section 9: Direct memory access controller \(DMA\) on page 182](#)) to the I2C\_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to [Master transmitter](#).
- 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](#) and [SMBus master transmitter](#).

*Note:* If DMA is used for transmission, the TXIE bit does not need to be enabled.

#### Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit in the I2C\_CR1 register. Data is loaded from the I2C\_RXDR register to an SRAM area configured using the DMA peripheral (refer to [Section 9: Direct memory access controller \(DMA\) on page 182](#)) 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 23.3](#)): the PEC transfer is managed with the NBYTES counter. Refer to [SMBus Slave receiver](#) and [SMBus master receiver](#).

*Note:* If DMA is used for reception, the RXIE bit does not need to be enabled.

### 23.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG\_I2Cx\_SMBUS\_TIMEOUT configuration bits in the DBG module.

## 23.5 I2C low-power modes

**Table 102. Effect of low-power modes on the I2C**

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop <sup>(1)</sup>	The I2C registers content is kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI48): the address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – 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 mode.

1. Refer to [Section 23.3](#) for information about the Stop modes supported by each instance. If wake-up from a specific Stop mode is not supported, the instance must be disabled before entering this Stop mode.

## 23.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 103. I2C Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
		Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
		Stop detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
		Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
		Transfer complete	TC		Write START = 1 or STOP = 1			
		Address matched	ADDR	ADDRIE	Write ADDRRCF = 1		Yes <sup>(1)</sup>	
		NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
I2C	I2C_ER	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
		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			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the Wake-up from Stop mode feature. Refer to [Section 23.3](#).

## 23.7 I2C registers

Refer to [Section 1.2 on page 35](#) for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

### 23.7.1 I2C control register 1 (I2C\_CR1)

Address offset: 0x000

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  $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$ .

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	SMBD EN	SMBH EN	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[3:0]				ERRIE	TCIE	STOP IE	NACK IE	ADDR IE	RXIE	TXIE	PE
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 **PECEN**: PEC enable

- 0: PEC calculation disabled
- 1: PEC calculation enabled

*Note:* If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).

Bit 22 **ALERTEN**: SMBus alert enable

- 0: The SMBus alert pin (SMBA) is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the Alert Response Address header is disabled (0001100x followed by NACK).
- 1: The SMBus alert pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the Alert Response Address header is enabled (0001100x followed by ACK).

*Note:* When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).*

Bit 21 **SMBDEN**: SMBus device default address enable

- 0: Device default address disabled. Address 0b1100001x is NACKed.
- 1: Device default address enabled. Address 0b1100001x is ACKed.

*Note:* If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).

Bit 20 **SMBHEN**: SMBus host address enable

- 0: Host address disabled. Address 0b0001000x is NACKed.
- 1: Host address enabled. Address 0b0001000x is ACKed.

*Note:* If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).

Bit 19 **GCEN**: General call enable

- 0: General call disabled. Address 0b00000000 is NACKed.
- 1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable

- 0: Wake-up from Stop mode disable.
- 1: Wake-up from Stop mode enable.

*Note: If the wake-up from Stop mode feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 23.3.*

*Note: WUPEN can be set only when DNF = '0000'*

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.

- 0: Clock stretching enabled
- 1: Clock stretching disabled

*Note: This bit can only be programmed when the I2C is disabled (PE = 0).*

Bit 16 **SBC**: Slave byte control

This bit is used to enable hardware byte control in slave mode.

- 0: Slave byte control disabled
- 1: Slave byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

- 0: DMA mode disabled for reception
- 1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

- 0: DMA mode disabled for transmission
- 1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **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, filters 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 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 three APB clock cycles.*

### 23.7.2 I2C control register 2 (I2C\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK1 + 6 x I2CCLK.

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	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD1 OR	ADD10	RD_ WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	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 in slave mode when SBC = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 23.3.*

Bit 25 **AUTOEND**: Automatic end mode (master mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

*Note: This bit has no effect in slave mode or when the RELOAD bit is set.*

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC = 0.

*Note: Changing these bits when the START bit is set is not allowed.*

**Bit 15 NACK:** NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

*Note: Writing '0' to this bit has no effect.*

*This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.*

*When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.*

*When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.*

**Bit 14 STOP:** Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

**In master mode:**

0: No Stop generation.

1: Stop generation after current byte transfer.

*Note: Writing '0' to this bit has no effect.*

**Bit 13 START:** Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by an address matched in slave mode, by a timeout error detection, or when PE = 0.

0: No Start generation.

**1: Restart/Start generation:**

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated start condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise setting this bit generates a START condition once the bus is free.

*Note: Writing '0' to this bit has no effect.*

*The START bit can be set even if the bus is BUSY or I2C is in slave mode.*

*This bit has no effect when RELOAD is set.*

**Bit 12 HEAD10R:** 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

*Note: Changing this bit when the START bit is set is not allowed.*

**Bit 11 ADD10:** 10-bit addressing mode (master mode)

0: The master operates in 7-bit addressing mode,

1: The master operates in 10-bit addressing mode

*Note: Changing this bit when the START bit is set is not allowed.*

**Bit 10 RD\_WRN:** Transfer direction (master mode)

0: Master requests a write transfer.

1: Master requests a read transfer.

*Note: Changing this bit when the START bit is set is not allowed.*

Bits 9:0 **SADD[9:0]**: Slave address (master mode)

**In 7-bit addressing mode (ADD10 = 0):**

SADD[7:1] must be written with the 7-bit slave address to be sent. The bits SADD[9], SADD[8] and SADD[0] are don't care.

**In 10-bit addressing mode (ADD10 = 1):**

SADD[9:0] must be written with the 10-bit slave address to be sent.

*Note: Changing these bits when the START bit is set is not allowed.*

### 23.7.3 I2C own address 1 register (I2C\_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$ .

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[9:0]									
rw					rw	rw	rw	rw	rw	rw	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 9:0 **OA1[9:0]**: Interface own slave address

7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. The bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address.

*Note: These bits can be written only when OA1EN = 0.*

### 23.7.4 I2C own address 2 register (I2C\_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access, until the previous

one is completed. The latency of the second write access can be up to 2x 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.	Res.	OA2MSK[2:0]			OA2[7:1]						Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	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

7-bit addressing mode: 7-bit address

*Note: These bits can be written only when OA2EN = 0.*

Bit 0 Reserved, must be kept at reset value.

### 23.7.5 I2C timing register (I2C\_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	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
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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](#)) and for SCL high and low level counters (refer to [I2C master initialization](#)).

$$t_{PRESC} = (\text{PRESC} + 1) \times t_{I2CCLK}$$

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 master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during  $t_{SCLDEL}$ .

$$t_{SCLDEL} = (\text{SCLDEL} + 1) \times t_{PRESC}$$

*Note:  $t_{SCLDEL}$  is used to generate  $t_{SU:DAT}$  timing.*

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay  $t_{SDADEL}$  between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during  $t_{SDADEL}$ .

$$t_{SDADEL} = \text{SDADEL} \times t_{PRESC}$$

*Note:  $t_{SDADEL}$  is used to generate  $t_{HD:DAT}$  timing.*

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

$$t_{SCLH} = (\text{SCLH} + 1) \times t_{PRESC}$$

*Note:  $t_{SCLH}$  is also used to generate  $t_{SU:STO}$  and  $t_{HD:STA}$  timing.*

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

$$t_{SCLL} = (\text{SCLL} + 1) \times t_{PRESC}$$

*Note:  $t_{SCLL}$  is also used to generate  $t_{BUF}$  and  $t_{SU:STA}$  timings.*

*Note:* This register must be configured when the I2C is disabled ( $PE = 0$ ).

*Note:* The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C Configuration window.

### 23.7.6 I2C timeout register (I2C\_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: No wait states, except if a write access occurs while a write access to this register is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times \text{PCLK1} + 6 \times \text{I2CCLK}$ .

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	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		
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]													
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	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 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

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

$t_{LOW:EXT} = (\text{TIMEOUTB} + \text{TIDLE} = 01) \times 2048 \times t_{I2CCLK}$

*Note: These bits can be written only when TEXTEN = 0.*

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled: when SCL is low for more than  $t_{TIMEOUT}$  (TIDLE = 0) or high for more than  $t_{IDLE}$  (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

*Note: This bit can be written only when TIMOUTEN = 0.*

Bits 11:0 **TIMEOUTA[11:0]**: Bus Timeout A

This field is used to configure:

The SCL low timeout condition  $t_{TIMEOUT}$  when TIDLE = 0

$t_{TIMEOUT} = (\text{TIMEOUTA} + 1) \times 2048 \times t_{I2CCLK}$

The bus idle condition (both SCL and SDA high) when TIDLE = 1

$t_{IDLE} = (\text{TIMEOUTA} + 1) \times 4 \times t_{I2CCLK}$

*Note: These bits can be written only when TIMOUTEN = 0.*

**Note:** If the SMBus feature is not supported, this register is reserved and forced by hardware to "0x00000000". Refer to [Section 23.3](#).

### 23.7.7 I2C interrupt and status register (I2C\_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	DIR
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]								
								r	r	r	r	r	r	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	rs	rs	

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

Bits 23:17 **ADDCODE[6:0]**: Address match code (slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1).

In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (slave mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. 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 an SMBALERT event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).*

Bit 12 **TIMEOUT**: Timeout or  $t_{LOW}$  detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).*

Bit 11 **PECERR**: PEC Error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

*If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).*

Bit 10 **OVR**: Overrun/Underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

**Bit 8 BERR:** Bus error

This flag is set by hardware when a misplaced Start or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting *BERRCF bit*.

*Note: This bit is cleared by hardware when PE = 0.*

**Bit 7 TCR:** Transfer Complete Reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

*Note: This bit is cleared by hardware when PE = 0.*

*This flag is only for master mode, or for slave mode when the SBC bit is set.*

**Bit 6 TC:** Transfer Complete (master mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

*Note: This bit is cleared by hardware when PE = 0.*

**Bit 5 STOPF:** Stop detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- either as a master, provided that the STOP condition is generated by the peripheral.
- or as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

**Bit 4 NACKF:** Not Acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

**Bit 3 ADDR:** Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting *ADDRCF bit*.

*Note: This bit is cleared by hardware when PE = 0.*

**Bit 2 RXNE:** Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C\_RXDR register, and is ready to be read. It is cleared when I2C\_RXDR is read.

*Note: This bit is cleared by hardware when PE = 0.*

**Bit 1 TXIS:** Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C\_TXDR register is empty and the data to be transmitted must be written in the I2C\_TXDR register. It is cleared when the next data to be sent is written in the I2C\_TXDR register.

This bit can be written to '1' by software 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 I2C\_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C\_TXDR register.

This bit can be written to '1' by software in order to flush the transmit data register I2C\_TXDR.

*Note: This bit is set by hardware when PE = 0.*

### 23.7.8 I2C interrupt clear register (I2C\_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: No wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	TIMOU TCF	PECCF	OVRCF	ARLOC F	BERRC F	Res.	Res.	STOPC F	NACKC F	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 I2C\_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).*

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C\_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).*

Bit 11 **PECCF**: PEC Error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C\_ISR register.

*Note: If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to [Section 23.3](#).*

Bit 10 **OVRCF**: Overrun/Underrun flag clear

Writing 1 to this bit clears the OVR flag in the I2C\_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear

Writing 1 to this bit clears the ARLO flag in the I2C\_ISR register.

Bit 8 **BERRCF**: Bus error flag clear

Writing 1 to this bit clears the BERRF flag in the I2C\_ISR register.

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

Bit 5 **STOPCF**: STOP detection flag clear

Writing 1 to this bit clears the STOPF flag in the I2C\_ISR register.

Bit 4 **NACKCF**: Not Acknowledge flag clear

Writing 1 to this bit clears the NACKF flag in I2C\_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear

Writing 1 to this bit clears the ADDR flag in the I2C\_ISR register. Writing 1 to this bit also clears the START bit in the I2C\_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

### 23.7.9 I2C PEC register (I2C\_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.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								PEC[7:0]							
								r	r	r	r	r	r	r	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". Refer to [Section 23.3](#).

### 23.7.10 I2C receive data register (I2C\_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.															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.								RXDATA[7:0]							
								r	r	r	r	r	r	r	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

### 23.7.11 I2C transmit data register (I2C\_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.	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.	Res.	Res.	Res.							
									rw						
TXDATA[7:0]															

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.

### 23.7.12 I2C register map

The table below provides the I2C register map and reset values.

Table 104. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x0	I2C_CR1	Res	Res	Res	Res	Res	Res	Res	Res	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
	Reset value								0	0	0	0	0	0	0	0	RXDMAEN
0x4	I2C_CR2	Res	Res	Res	Res	Res	Res	Res	PECBYTE	0	0	0	0	0	0	0	TXDMAEN
	Reset value								0	0	0	0	0	0	0	0	SADD[9:0]
0x8	I2C_OAR1	Res	Res	Res	Res	Res	Res	Res	NBYTES[7:0]	0	0	0	0	0	0	0	DNF[3:0]
	Reset value								NACK	0	0	0	0	0	0	0	ERRIE
0xC	I2C_OAR2	Res	Res	Res	Res	Res	Res	Res	STOP	0	0	0	0	0	0	0	TCIE
	Reset value								START	0	0	0	0	0	0	0	STOPIE
0x10	I2C_TIMINGR	PRESC[3:0]						SCLDEL [3:0]	SDADEL [3:0]	0	0	0	0	0	0	0	OA1MODE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	OA2MS K [2:0]

**Table 104. I2C register map and reset values (continued)**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x14	I2C_TIMEOUTR	TEXten																																
	Reset value	0																																
0x18	I2C_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.		
	Reset value																																	
0x1C	I2C_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.		
	Reset value																																	
0x20	I2C_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.		
	Reset value																																	
0x24	I2C_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.		
	Reset value																																	
0x28	I2C_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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																	

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 24 Universal synchronous receiver transmitter (USART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

### 24.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and Half-duplex Single-wire communications, as well as LIN (local interconnection network), Smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and Modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

## 24.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data
  - Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire Half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from Mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

## 24.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for Smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

## 24.4 USART implementation

The table(s) below describe(s) USART implementation. It(they) also include(s) LPUART for comparison.

**Table 105. STM32C0x1 features**

USART instances	STM32C0x1
USART1	FULL
USART2	BASIC

**Table 106. USART features**

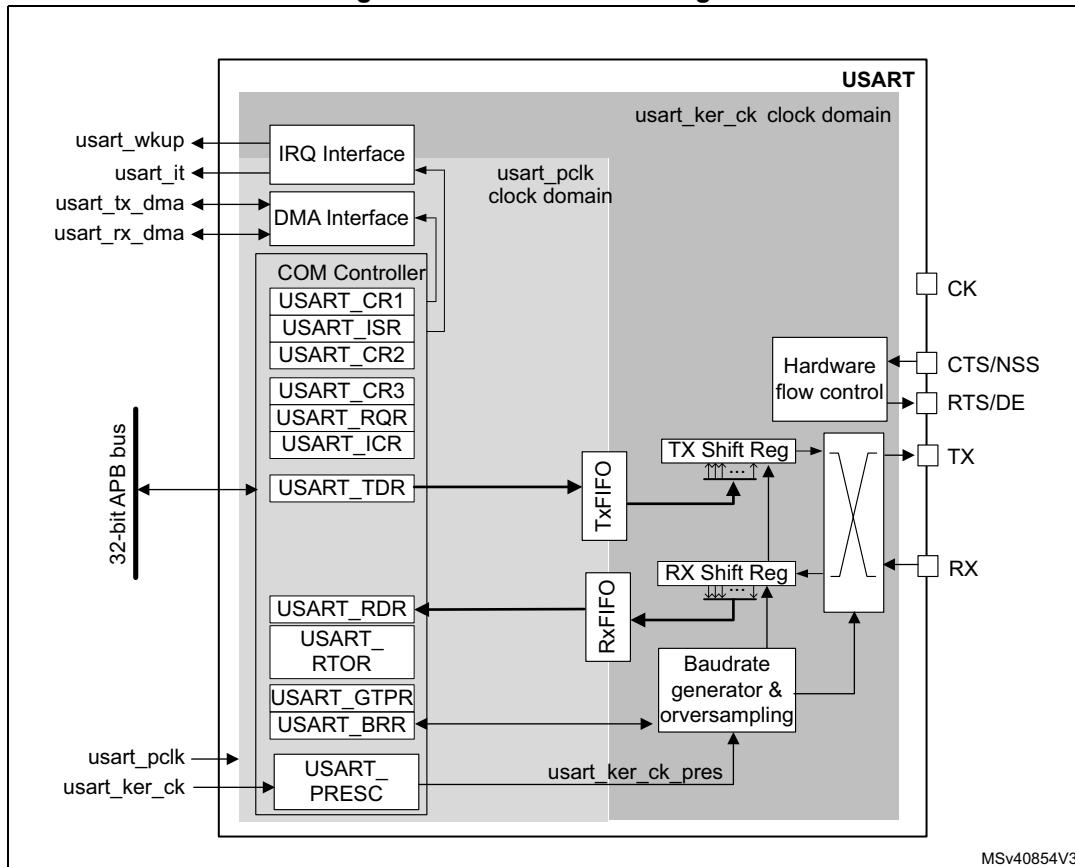
USART modes/features <sup>(1)</sup>	Full feature set	Basic feature set
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	X
Smartcard mode	X	-
Single-wire Half-duplex communication	X	X
IrDA SIR ENDEC block	X	-
LIN mode	X	-
Dual clock domain and wakeup from low-power mode	X	-
Receiver timeout interrupt	X	-
Modbus communication	X	-
Auto baud rate detection	X	-
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	-
Tx/Rx FIFO size	8	-
Prescaler	X	-

1. X = supported.

## 24.5 USART functional description

### 24.5.1 USART block diagram

Figure 241. USART block diagram



The simplified block diagram given in [Figure 241](#) shows two fully-independent clock domains:

- The **usart\_pclk** clock domain  
The **usart\_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
  - The **usart\_ker\_ck** kernel clock domain.  
The **usart\_ker\_ck** is the USART clock source. It is independent from **usart\_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart\_ker\_ck** clock is stopped.
- When the dual clock domain feature is disabled, the **usart\_ker\_ck** clock is the same as the **usart\_pclk** clock.

There is no constraint between **usart\_pclk** and **usart\_ker\_ck**: **usart\_ker\_ck** can be faster or slower than **usart\_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart\_ker\_ck** clock must be at least 3 times faster than the clock on the CK input.

## 24.5.2 USART signals

### USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

- **TX** (Transmit Data Output)

When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In Single-wire and Smartcard modes, this I/O is used to transmit and receive data.

### RS232 Hardware flow control mode

The following pins are required in RS232 Hardware flow control mode:

- **CTS** (Clear To Send)

When driven high, this signal blocks the data transmission at the end of the current transfer.

- **RTS** (Request To Send)

When it is low, this signal indicates that the USART is ready to receive data.

### RS485 Hardware control mode

The following pin is required in RS485 Hardware control mode:

- **DE** (Driver Enable)

This signal activates the transmission mode of the external transceiver.

*Note:* DE and RTS share the same pin.

### Synchronous master/slave mode and Smartcard mode

The following pin is required in synchronous master/slave mode and Smartcard mode:

- **CK**

This pin acts as Clock output in Synchronous master and Smartcard modes.

It acts as Clock input in Synchronous slave mode.

In Synchronous Master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

In Smartcard mode, CK output provides the clock to the smartcard.

- **NSS**

This pin acts as Slave Select input in Synchronous slave mode.

*Note:* NSS and CTS share the same pin.

### 24.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART\_CR1 register (see [Figure 242](#)):

- 7-bit character length: M[1:0] = '10'
- 8-bit character length: M[1:0] = '00'
- 9-bit character length: M[1:0] = '01'

**Note:** *In 7-bit data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

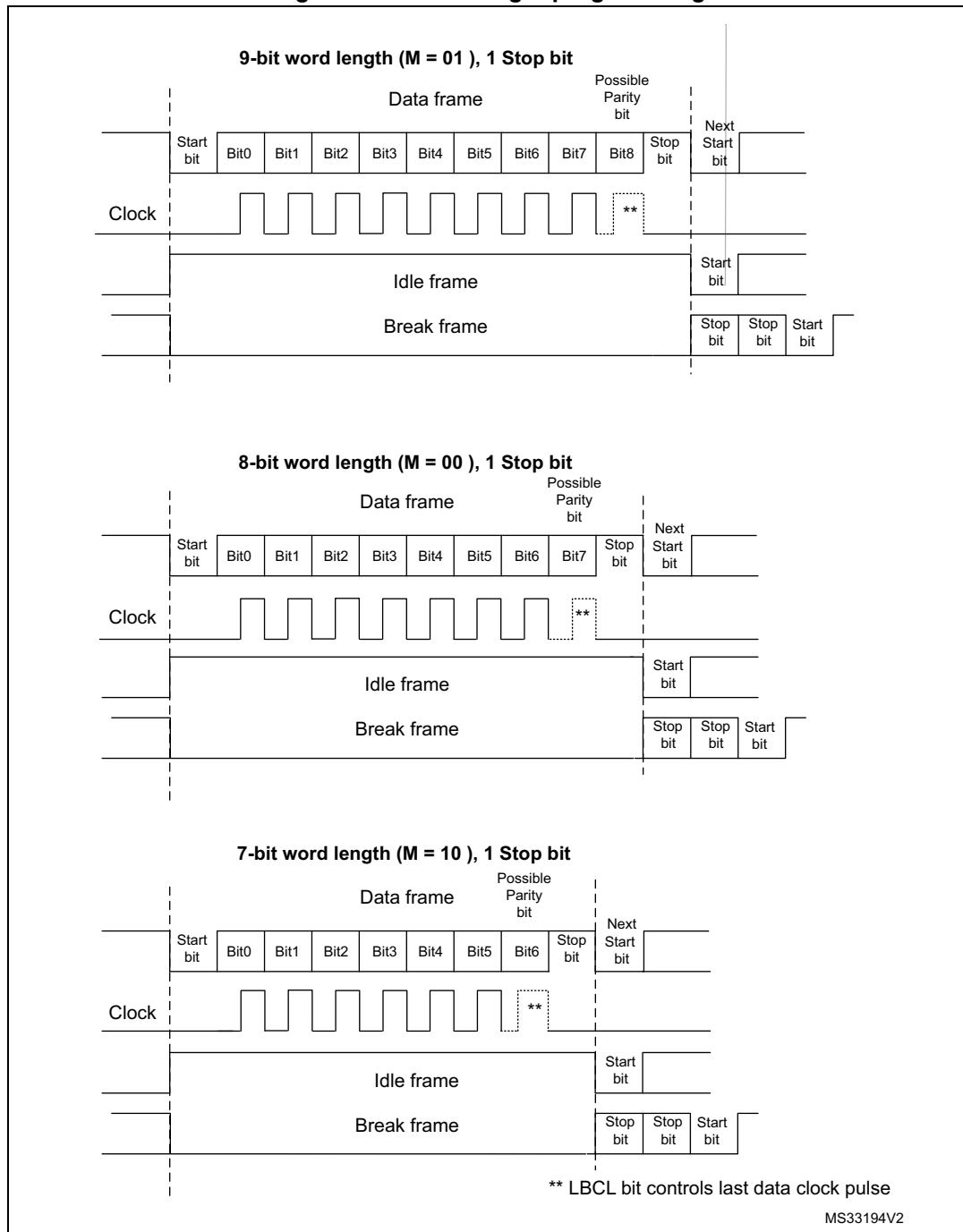
An **Idle character** is interpreted as an entire frame of "1"s (the number of "1"s includes the number of stop bits).

A **Break character** is interpreted on receiving "0"s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 242. Word length programming



#### 24.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART\_CR1 register (bit 29). This mode is supported only in UART, SPI and Smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

*Note:* *The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.*

*The status flags are available in the USART\_ISR register.*

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART\_CR3 control register.

In this case:

- The RXFT flag is set in the USART\_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

RXFTCFG data have been received: one data in USART\_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART\_RDR). As a result, the next received data is not set the overrun flag.

- The TXFT flag is set in the USART\_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

#### 24.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

##### Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART\_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART\_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

**Note:** *The TE bit must be set before writing the data to be transmitted to the USART\_TDR.*

*The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.*

*An idle frame is sent when the TE bit is enabled.*

#### Configurable stop bits

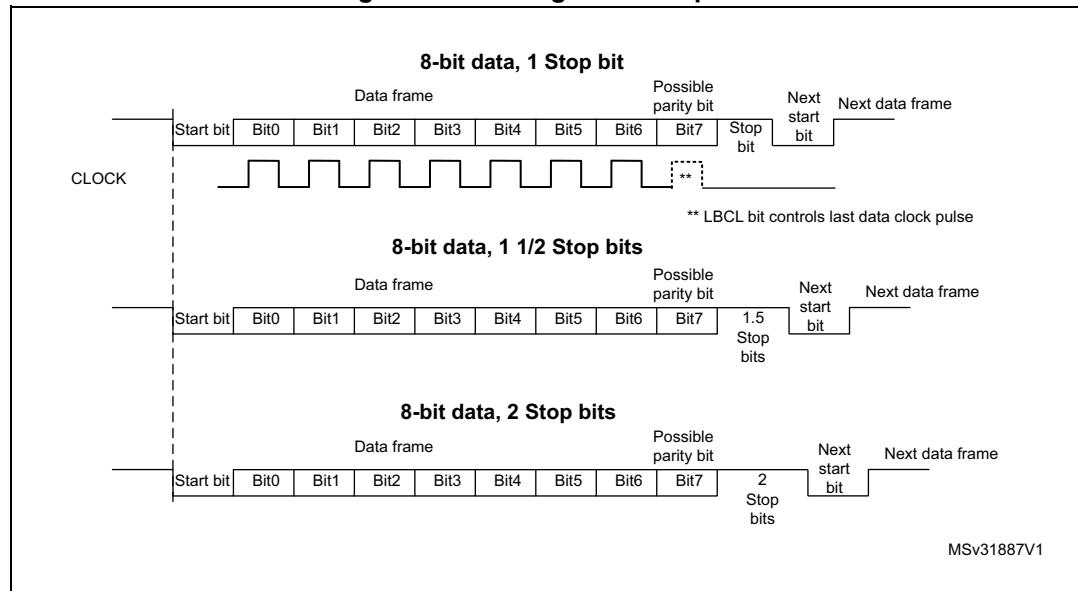
The number of stop bits to be transmitted with every character can be programmed in USART\_CR2, bits 13,12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This is supported by normal USART, Single-wire and Modem modes.
- **1.5 stop bits:** To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits (see [Figure 243](#)). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 243. Configurable stop bits**



### Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART\_CR1 to define the word length.
2. Select the desired baud rate using the USART\_BRR register.
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to 1.
5. Select DMA enable (DMAT) in USART\_CR3 if multibuffer communication must take place. Configure the DMA register as explained in [Section 24.5.19: Continuous communication using USART and DMA](#).
6. Set the TE bit in USART\_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART\_TDR register. Repeat this for each data to be transmitted in case of single buffer.
  - When FIFO mode is disabled, writing a data to the USART\_TDR clears the TXE flag.
  - When FIFO mode is enabled, writing a data to the USART\_TDR adds one data to the TXFIFO. Write operations to the USART\_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART\_TDR register, wait until TC = 1.
  - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
  - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

## Single byte communication

- When FIFO mode is disabled

Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:

- the data have been moved from the USART\_TDR register to the shift register and the data transmission has started;
- the USART\_TDR register is empty;
- the next data can be written to the USART\_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is ongoing, a write instruction to the USART\_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

When no transmission is ongoing, a write instruction to the USART\_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:

- the TXFIFO is not full;
- the USART\_TDR register is empty;
- the next data can be written to the USART\_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART\_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

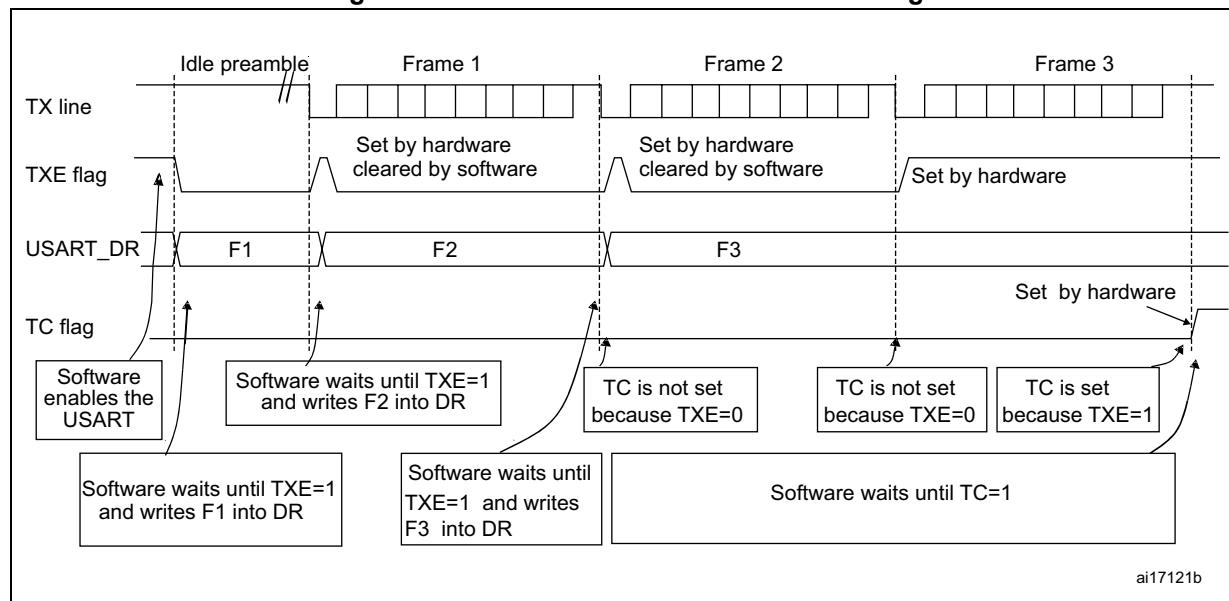
When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write operation to USART\_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART\_CR1 register.

After writing the last data to the USART\_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the device to enter the low-power mode (see [Figure 244: TC/TXE behavior when transmitting](#)).

Figure 244. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

### Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see [Figure 242](#)).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

### Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

## 24.5.6 USART receiver

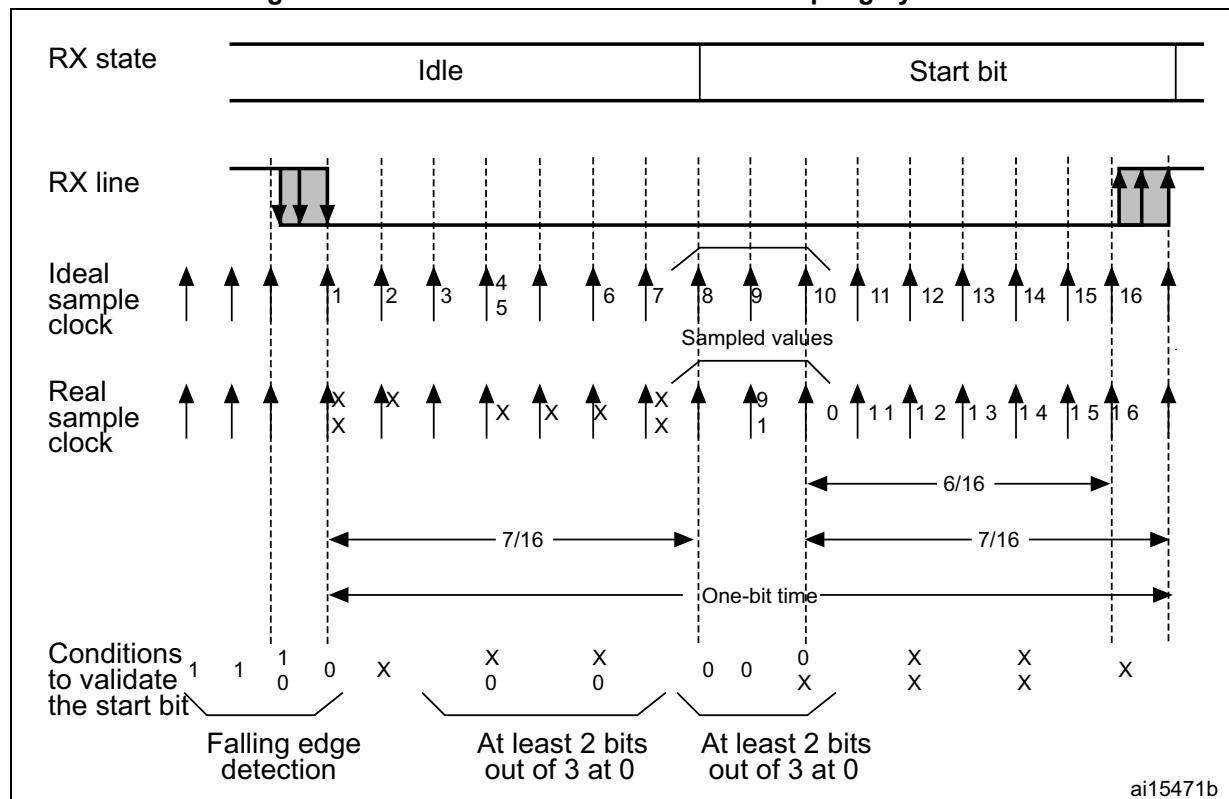
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART\_CR1 register.

### Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0 X 0 X 0 X 0 X 0.

Figure 245. 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 and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at '0' (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at '0' and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at '0').

The start bit is validated but the NE noise flag is set if,

- 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
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at '0'.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

## Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

### Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART\_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART\_BRR
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to '1'.
5. Select DMA enable (DMAR) in USART\_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 24.5.19: Continuous communication using USART and DMA](#).
6. Set the RE bit USART\_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART\_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART\_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to '1' in the USART\_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART\_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to '1' in USART\_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

### Break character

When a break character is received, the USART handles it as a framing error.

### Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

### Overrun error

- FIFO mode disabled

An overrun error occurs if a character is received and RXNE has not been reset.

Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- the ORE bit is set;
- the RDR content is not lost. The previous data is available by reading the USART\_RDR register.
- the shift register is overwritten. After that, any data received during overrun is lost.
- an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

Data can not be transferred from the shift register to the USART\_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

- The ORE bit is set.
- The first entry in the RXFIFO is not lost. It is available by reading the USART\_RDR register.
- The shift register is overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART\_ICR register.

*Note:*

*The ORE bit, when set, indicates that at least 1 data has been lost.*

*When the FIFO mode is disabled, there are two possibilities*

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

### Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section *Reset and clock control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

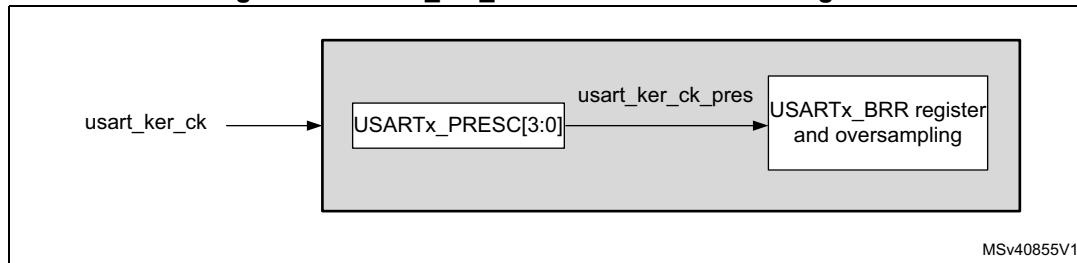
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is `usart_ker_ck`.

When the dual clock domain and the wake-up from low-power mode features are supported, the `usart_ker_ck` clock source can be configurable in the RCC (see Section *Reset and clock control (RCC)*). Otherwise the `usart_ker_ck` clock is the same as `usart_pclk`.

The `usart_ker_ck` clock can be divided by a programmable factor, defined in the `USART_PRESC` register.

**Figure 246. usart\_ker\_ck clock divider block diagram**



Some `usart_ker_ck` sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the `USART_RDR` register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (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 enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the `OVER8` bit in the `USART_CR1` register either to 16 or 8 times the baud rate clock (see [Figure 247](#) and [Figure 248](#)).

Depending on your application:

- select oversampling by 8 (`OVER8 = 1`) to achieve higher speed (up to `usart_ker_ck_pres/8`). In this case the maximum receiver tolerance to clock deviation is reduced (refer to [Section 24.5.8: Tolerance of the USART receiver to clock deviation on page 678](#))
- 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

`uart_ker_ck_pres/16` (where `uart_ker_ck_pres` is the USART input clock divided by a prescaler).

Programming the `ONEBIT` bit in the `USART_CR3` register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the `NE` bit is set.
- A single sample in the center of the received bit

Depending on your application:

- select the three sample majority vote method (`ONEBIT = 0`) when operating in a noisy environment and reject the data when a noise is detected (refer to [Figure 107](#)) because this indicates that a glitch occurred during the sampling.
- select the single sample method (`ONEBIT = 1`) when the line is noise-free to increase the receiver tolerance to clock deviations (see [Section 24.5.8: Tolerance of the USART receiver to clock deviation on page 678](#)). In this case the `NE` bit is never set.

When noise is detected in a frame:

- The `NE` bit is set at the rising edge of the `RXNE` bit (`RXFNE` in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the `USART_RDR` register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the `RXNE` bit (`RXFNE` in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the `EIE` bit is set in the `USART_CR3` register.

The `NE` bit is reset by setting `NECF` bit in `USART_ICR` register.

Note:

*Noise error is not supported in SPI mode.*

*Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes. In those modes, the `OVER8` bit is forced to '0' by hardware.*

**Figure 247. Data sampling when oversampling by 16**

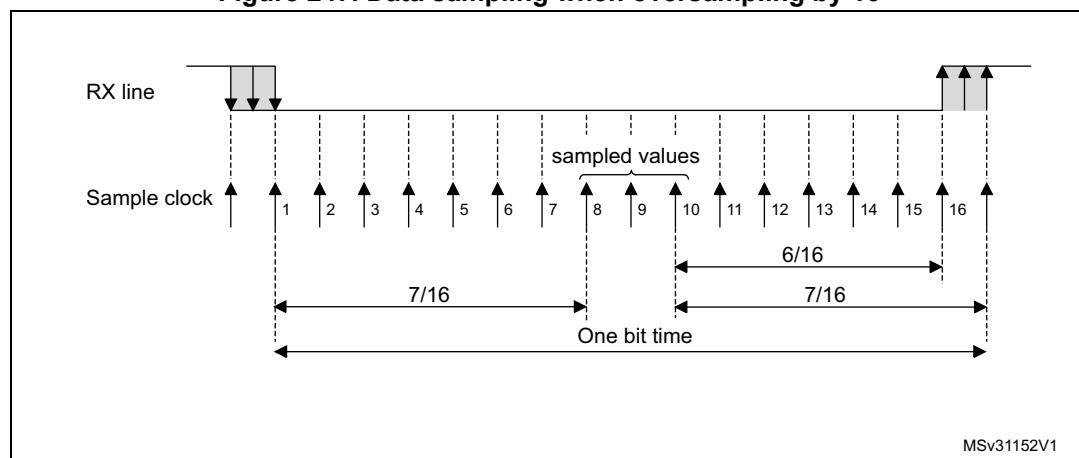


Figure 248. Data sampling when oversampling by 8

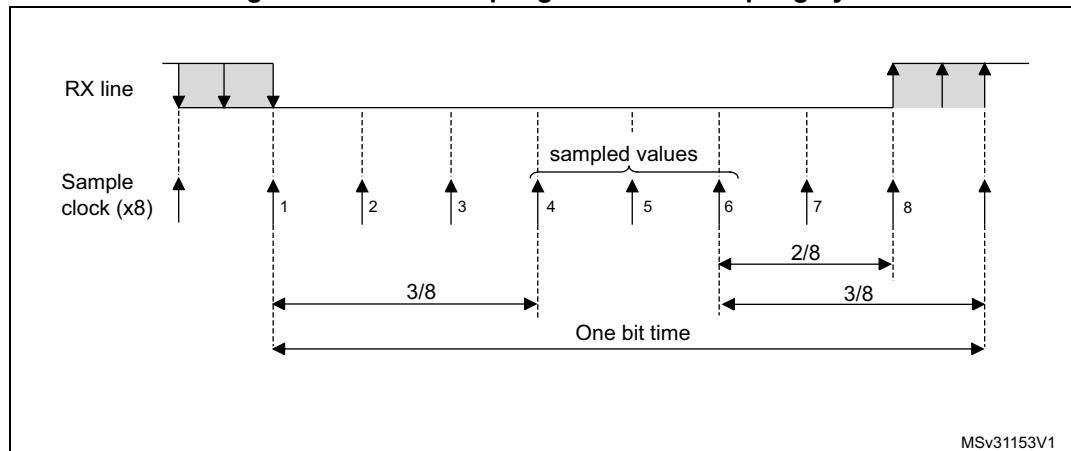


Table 107. Noise detection from sampled data

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 (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART\_CR3 register.

The FE bit is reset by writing '1' to the FECF in the USART\_ICR register.

*Note:* Framing error is not supported in SPI mode.

### Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART\_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

- **0.5 stop bit (reception in Smartcard mode):** no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (Smartcard mode)**

When transmitting in Smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART\_CR1) and the stop bit is checked to test if the Smartcard has detected a parity error.

In the event of a parity error, the Smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to [Section 24.5.16: USART receiver timeout on page 692](#) for more details).

- **2 stop bits**

Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

The framing error flag is set if a framing error is detected during the first stop bit.

The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

### 24.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART\_BRR register.

#### Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = '0' or '1')

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{uart\_ker\_ckpres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{uart\_ker\_ckpres}}{\text{USARTDIV}}$$

#### Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{uart\_ker\_ckpres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART\_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] 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 and 8, USARTDIV must be greater than or equal to 16.*

### How to derive USARTDIV from USART\_BRR register values

#### Example 1

To obtain 9600 baud with usart\_ker\_ck\_pres = 8 MHz:

- In case of oversampling by 16:  
USARTDIV = 8 000 000/9600  
BRR = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:  
USARTDIV = 2 \* 8 000 000/9600  
USARTDIV = 1666,66 (0d1667 = 0x683)  
BRR[3:0] = 0x3 >> 1 = 0x1  
BRR = 0x681

#### Example 2

To obtain 921.6 Kbaud with usart\_ker\_ck\_pres = 48 MHz:

- In case of oversampling by 16:  
USARTDIV = 48 000 000/921 600  
BRR = USARTDIV = 0d52 = 0x34
- In case of oversampling by 8:  
USARTDIV = 2 \* 48 000 000/921 600  
USARTDIV = 104 (0d104 = 0x68)  
BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4  
BRR = 0x64

### 24.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$  is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in [Table 108](#), [Table 109](#), depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART\_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART\_CR1 register
- Bits BRR[3:0] of USART\_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART\_CR3 register.

**Table 108. Tolerance of the USART receiver when BRR [3:0] = 0000**

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

**Table 109. Tolerance of the USART receiver when BRR[3:0] is different from 0000**

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

**Note:** The data specified in [Table 108](#) and [Table 109](#) may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9-bit times when M = 10).

#### 24.5.9 USART auto baud rate detection

The USART can detect and automatically set the USART\_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart\_ker\_ck\_pres/65535 and usart\_ker\_ck\_pres/16.
- When oversampling by 8, the baud rate ranges from usart\_ker\_ck\_pres/65535 and usart\_ker\_ck\_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART\_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0:** Any character starting with a bit at '1'.  
In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.  
In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2:** A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3:** A 0x55 character frame.  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART\_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART\_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART\_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a '0').

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART\_ISR register.

**Note:** *The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.*

#### 24.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in Mute mode by means of the muting function. To use the Mute mode feature, the MME bit must be set in the USART\_CR1 register.

**Note:** *When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart\_ker\_ck cycles), otherwise Mute mode might remain active.*

When the Mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART\_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART\_RQR register, under certain conditions.

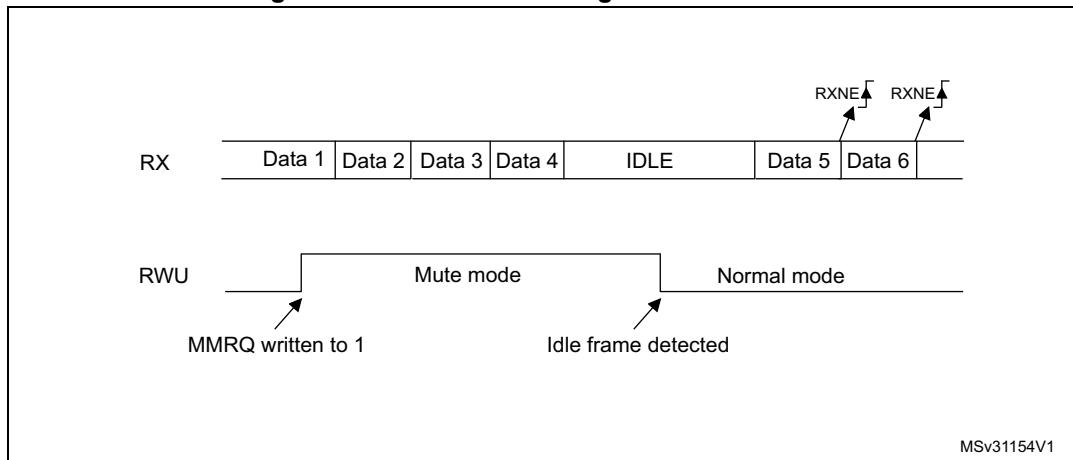
The USART can enter or exit from Mute mode using one of two methods, depending on the WAKE bit in the USART\_CR1 register:

- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

##### Idle line detection (WAKE = 0)

The USART enters Mute mode when the MMRQ bit is written to '1' and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART\_ISR register. An example of Mute mode behavior using Idle line detection is given in [Figure 249](#).

**Figure 249. Mute mode using Idle line detection**

**Note:** If the MMRQ is set while the IDLE character has already elapsed, Mute mode is not entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

#### 4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a '1', otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART\_CR2 register.

**Note:** In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

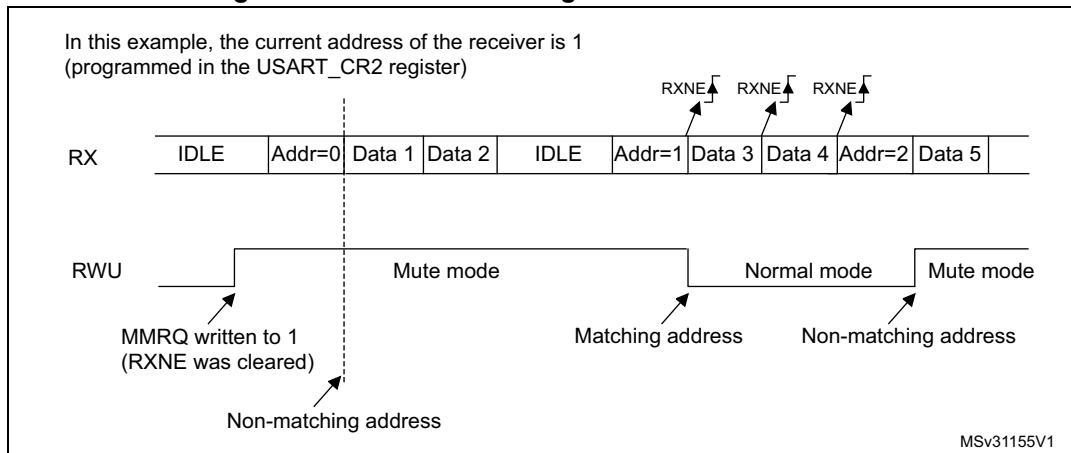
The USART enters Mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters Mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering Mute mode.

The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from Mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

**Note:** When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in Mute mode

An example of Mute mode behavior using address mark detection is given in [Figure 250](#).

**Figure 250. Mute mode using address mark detection**

#### 24.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a Half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

##### Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART\_CR2 register and the RTOIE in the USART\_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

##### Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

### 24.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART\_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in [Table 110](#).

**Table 110. USART frame formats**

M bits	PCE bit	USART frame <sup>(1)</sup>
00	0	SB   8 bit data   STB
00	1	SB   7-bit data   PB   STB
01	0	SB   9-bit data   STB
01	1	SB   8-bit data PB   STB
10	0	SB   7bit data   STB
10	1	SB   6-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 (8th or 7th, depending on the M bit value).

#### Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits are set, the parity bit is equal to 0 if even parity is selected (PS bit in USART\_CR1 = 0).

#### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART\_CR1 = 1).

#### Parity checking in reception

If the parity check fails, the PE flag is set in the USART\_ISR register and an interrupt is generated if PEIE is set in the USART\_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART\_ICR register.

#### Parity generation in transmission

If the PCE bit is set in USART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS=1)).

### 24.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to [Section 24.4: USART implementation on page 660](#).

The LIN mode is selected by setting the LINEN bit in the USART\_CR2 register. In LIN mode, the following bits must be kept cleared:

- CLKEN in the USART\_CR2 register,
- STOP[1:0], SCEN, HDSEL and IREN in the USART\_CR3 register.

#### LIN transmission

The procedure described in [Section 24.5.4](#) has to be applied for LIN Master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 '0' bits as a break character. Then two bits of value '1' are sent to enable the next start detection.

#### LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART\_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART\_CR2) or 11 (when LBDL = 1 in USART\_CR2) consecutive bits are detected as '0, and are followed by a delimiter character, the LBDF flag is set in USART\_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

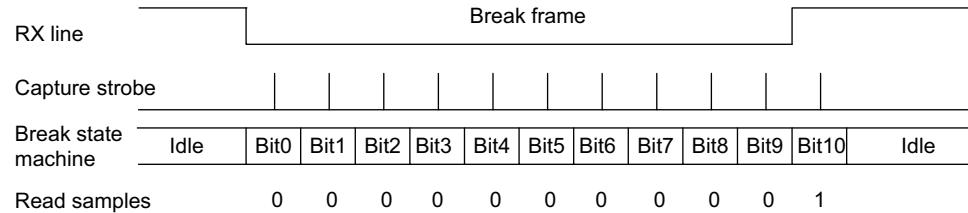
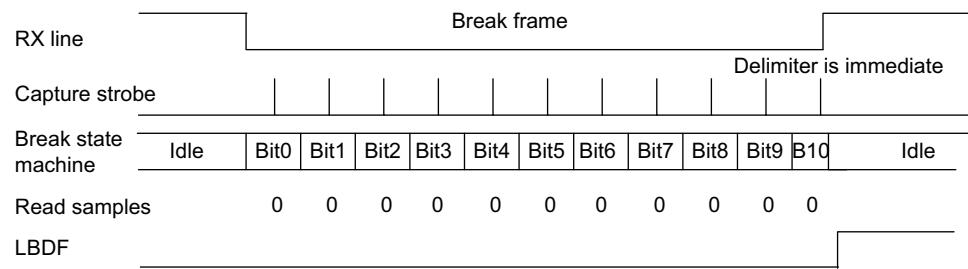
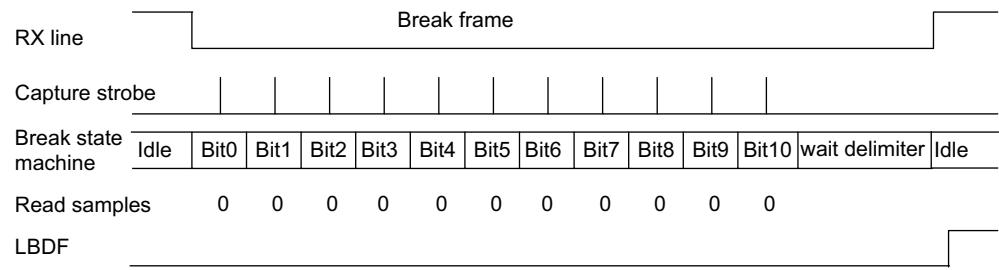
If a '1' is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

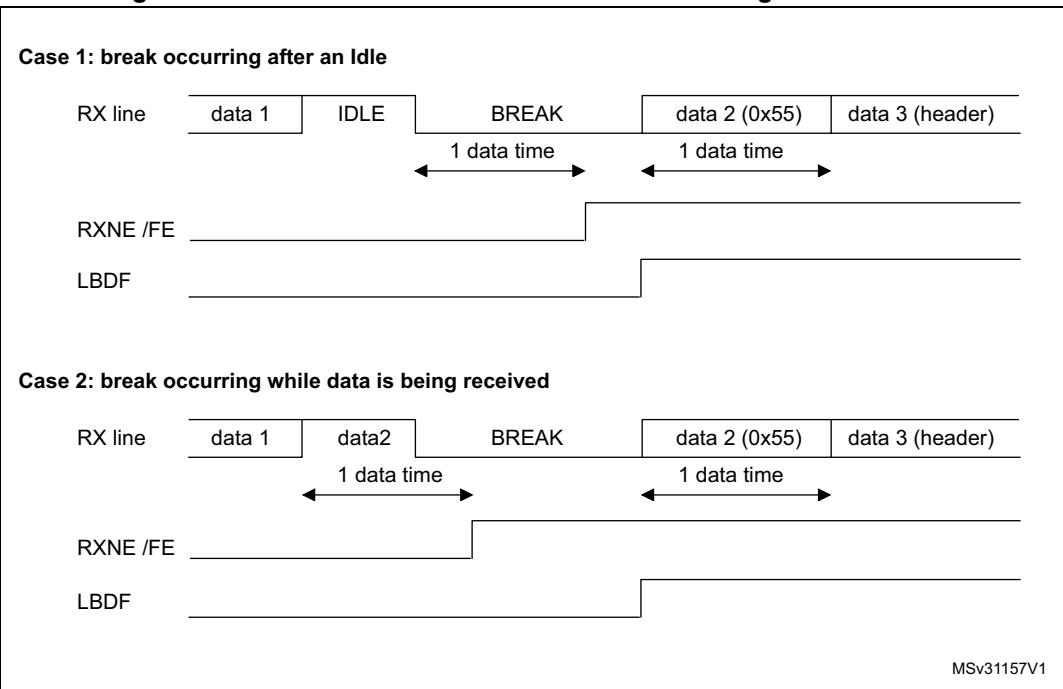
If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (i.e. stop bit detected at '0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a '1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the [Figure 251: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 687](#).

Examples of break frames are given on [Figure 252: Break detection in LIN mode vs. Framing error detection on page 688](#).

**Figure 251. Break detection in LIN mode (11-bit break length - LBDL bit is set)****Case 1: break signal not long enough => break discarded, LBDF is not set****Case 2: break signal just long enough => break detected, LBDF is set****Case 3: break signal long enough => break detected, LBDF is set**

MSv31156V1

**Figure 252. Break detection in LIN mode vs. Framing error detection**

#### 24.5.14 USART synchronous mode

##### Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART\_CR2 register to '1'. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART\_CR2 register,
- SCEN, HDSEL and IREN bits in the USART\_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART\_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART\_CR2 register is used to select the clock polarity, and the CPHA bit in the USART\_CR2 register is used to select the phase of the external clock (see [Figure 253](#), [Figure 254](#) and [Figure 255](#)).

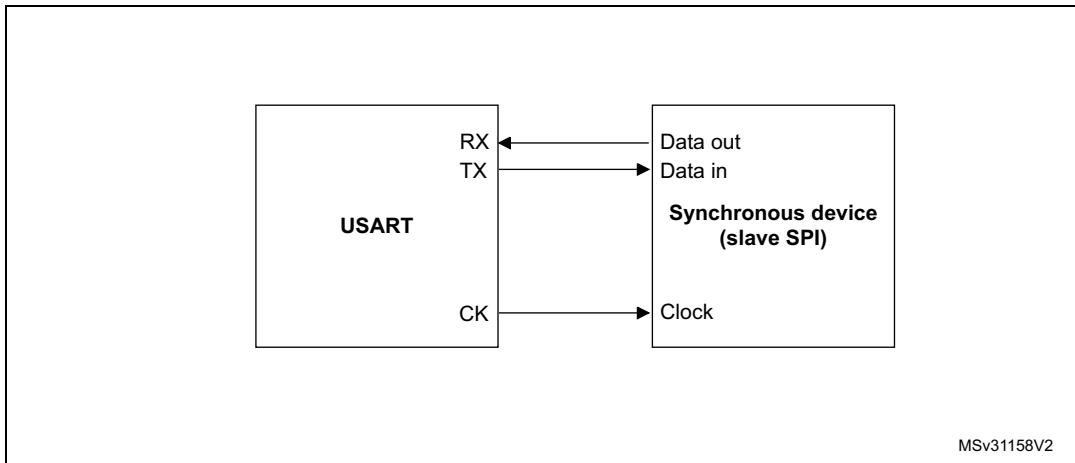
During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

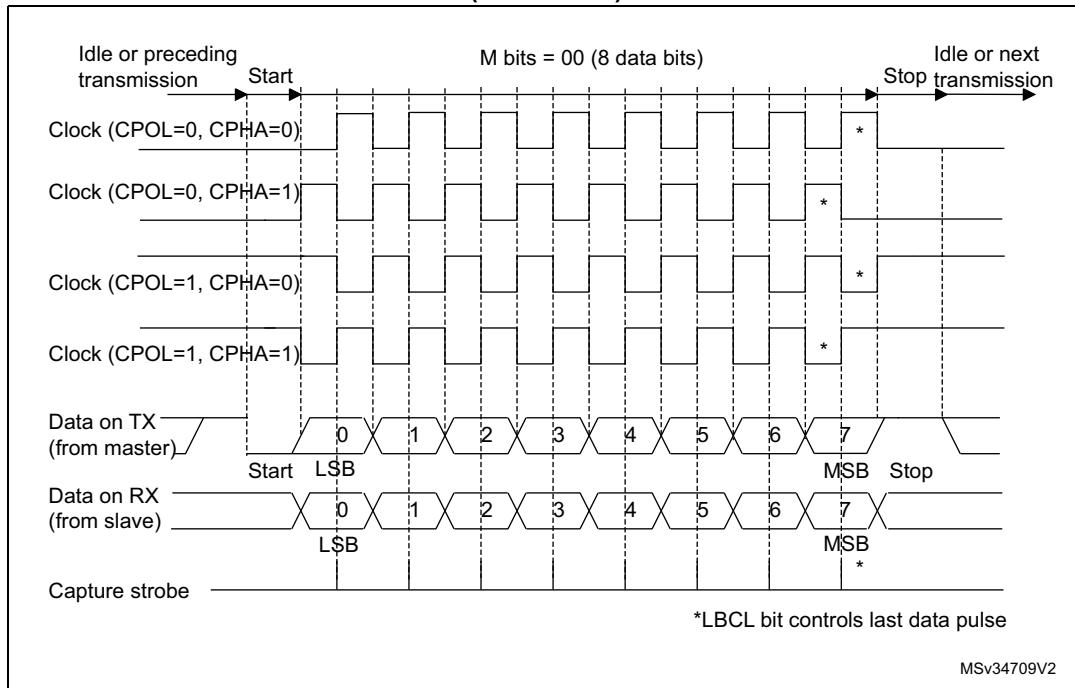
In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

**Note:** In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled ( $TE = 1$ ) and data are being transmitted (USART\_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

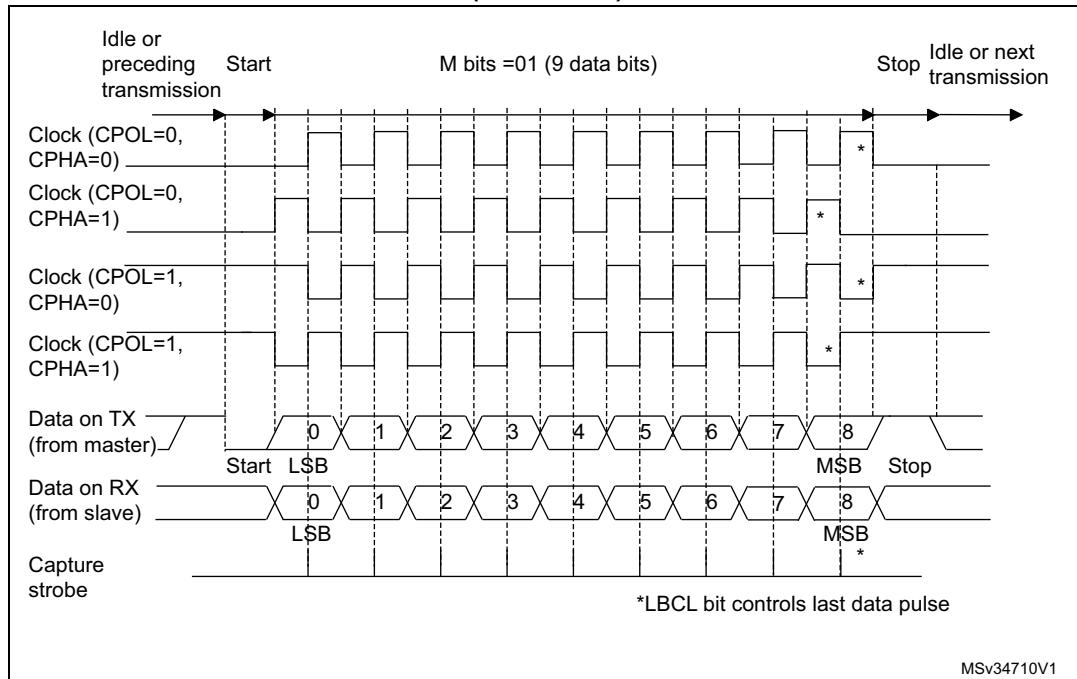
**Figure 253. USART example of synchronous master transmission**



**Figure 254. USART data clock timing diagram in synchronous master mode (M bits = 00)**



**Figure 255. USART data clock timing diagram in synchronous master mode  
(M bits = 01)**



MSv34710V1

### Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART\_CR2 register to '1'. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART\_CR2 register,
- SCEN, HDSEL and IREN bits in the USART\_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

*Note:*

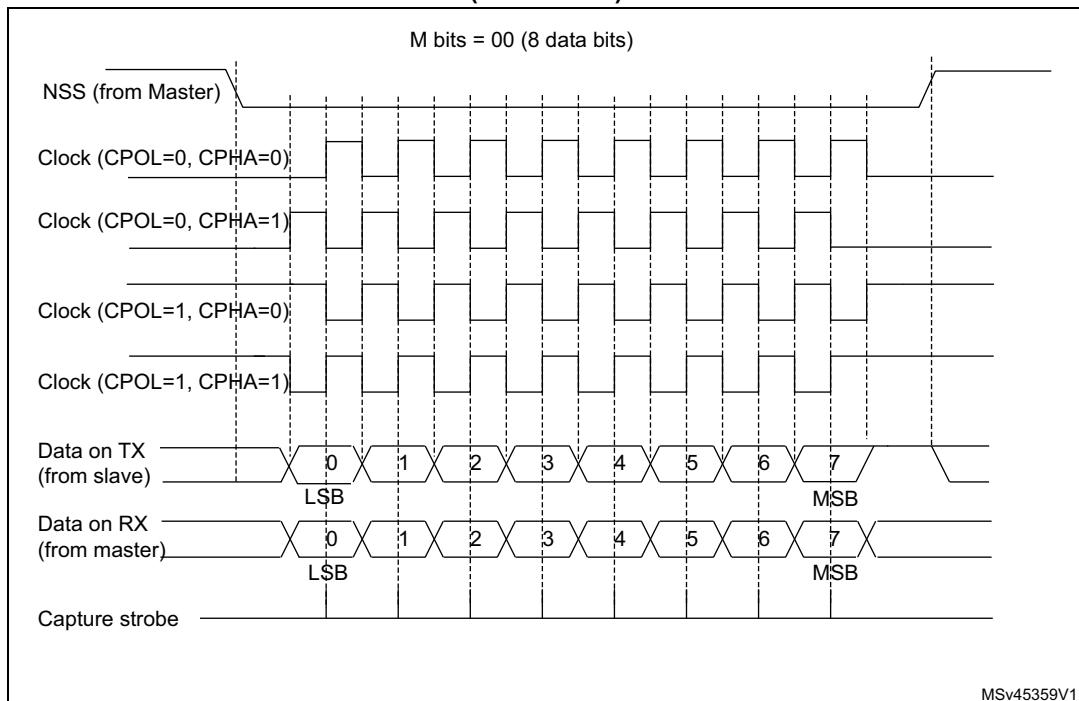
*When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (uart\_ker\_ck\_pres) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART\_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see [Figure 256](#)).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART\_TDR.

The slave supports the hardware and software NSS management.

**Figure 256. USART data clock timing diagram in synchronous slave mode  
(M bits = 00)**



### Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS\_NSS bit in the USART\_CR2 register:

- Software NSS management (DIS\_NSS = 1)
  - The SPI slave is always selected and NSS input pin is ignored.
  - The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS\_NSS = 0)
  - The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

*Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.*

*In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.*

### SPI Slave underrun error

When an underrun error occurs, the UDR flag is set in the USART\_ISR register, and the SPI slave goes on sending the last data until the underrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART\_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART\_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

**Note:** *An underrun error may occur if the moment the data is written to the USART\_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART\_TDR should be written 3 usart\_ker\_ck cycles before the first CK edge.*

#### 24.5.15 USART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit in the USART\_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART\_CR2 register,
- SCEN and IREN bits in the USART\_CR3 register.

The USART can be configured to follow a Single-wire Half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART\_CR3.

As soon as HDSEL is written to '1':

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

#### 24.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART\_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART\_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = '00' or STOP = '11'
- from the end of the second stop bit if STOP = '10'.
- from the beginning of the stop bit if STOP = '01'.

When the timeout duration has elapsed, the RTOF flag in the USART\_ISR register is set. A timeout is generated if RTOIE bit in USART\_CR1 register is set.

### 24.5.17 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to [Section 24.4: USART implementation on page 660](#).

Smartcard mode is selected by setting the SCEN bit in the USART\_CR3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART\_CR2 register,
- HDSEL and IREN bits in the USART\_CR3 register.

The CLKEN bit can also be set to provide a clock to the Smartcard.

The Smartcard interface is designed to support asynchronous Smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

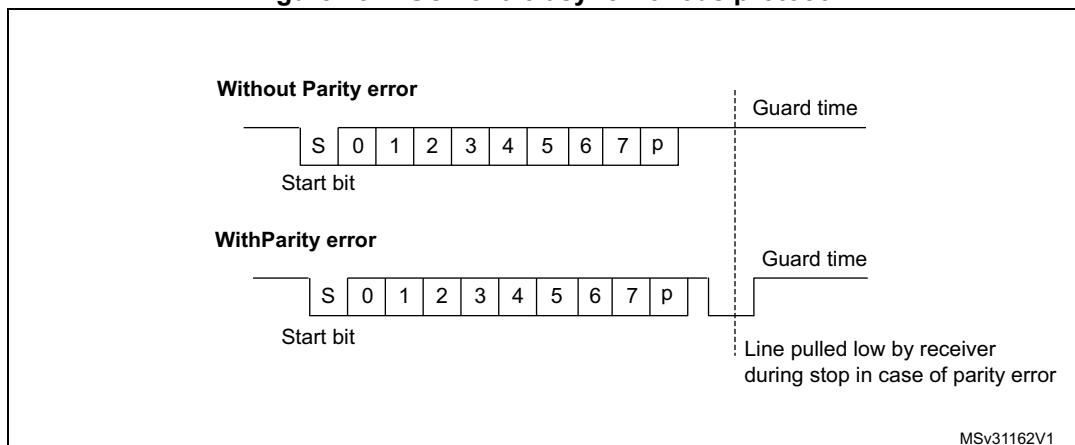
The USART should be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART\_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = '11' in the USART\_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

[Figure 257](#) shows examples of what can be seen on the data line with and without parity error.

**Figure 257. ISO 7816-3 asynchronous protocol**



When connected to a Smartcard, the TX output of the USART drives a bidirectional line that is also driven by the Smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the Smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART\_RQR register.

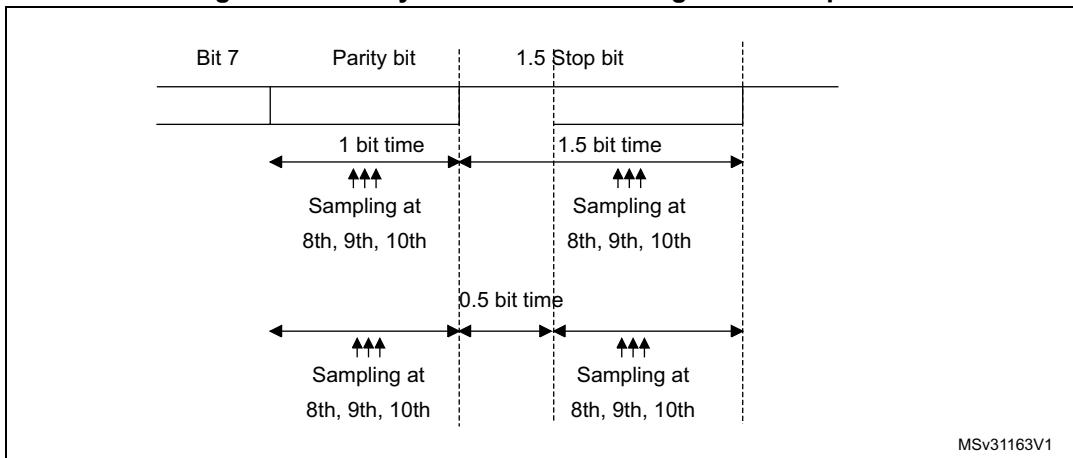
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the Smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The deassertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

*Note:*

*Break characters are not significant in Smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.*

*No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.*

*Figure 258* shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

**Figure 258. Parity error detection using the 1.5 stop bits**

The USART can provide a clock to the Smartcard through the CK output. In Smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART\_GTPR register. CK frequency can be programmed from `usart_ker_ck_pres/2` to `usart_ker_ck_pres/62`, where `usart_ker_ck_pres` is the peripheral input clock divided by a programmed prescaler.

### Block mode ( $T = 1$ )

In  $T = 1$  (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART\_CR3 register.

When requesting a read from the Smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

**Note:** *The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the Smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.*

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the Smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

**Note:** *As in the Smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.*

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the Smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART\_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN = LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN = LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBI bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

*Note:* *The error checking code (LRC/CRC) must be computed/verified by software.*

### Direct and inverse convention

The Smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV = 1.

*Note:* *When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.*

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, assuming that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH results in a USART received character of 03 and an odd parity.

Therefore, two methods are available for TS pattern recognition:

#### Method 1

The USART is programmed in standard Smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART.

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

#### Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

- (H) LHHL LLL LLH = 0x103: inverse convention to be chosen
- (H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

### 24.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to [Section 24.4: USART implementation on page 660](#).

IrDA mode is selected by setting the IREN bit in the USART\_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART\_CR2 register,
- SCEN and HDSEL bits in the USART\_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see [Figure 259](#)).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbaud for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not

encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A ‘0’ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see [Figure 260](#)).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41  $\mu$ 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). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART\_CR2 register must be configured to ‘1 stop bit’.

### IrDA low-power mode

- Transmitter

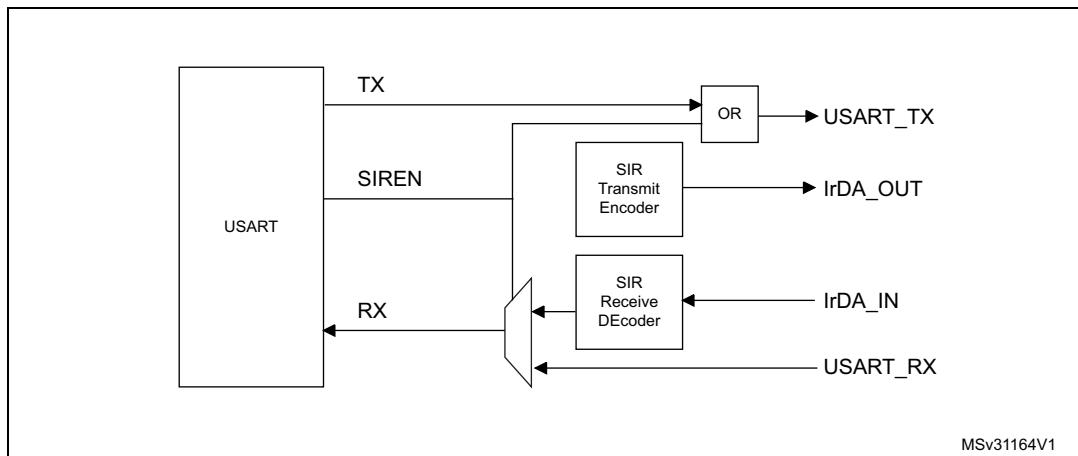
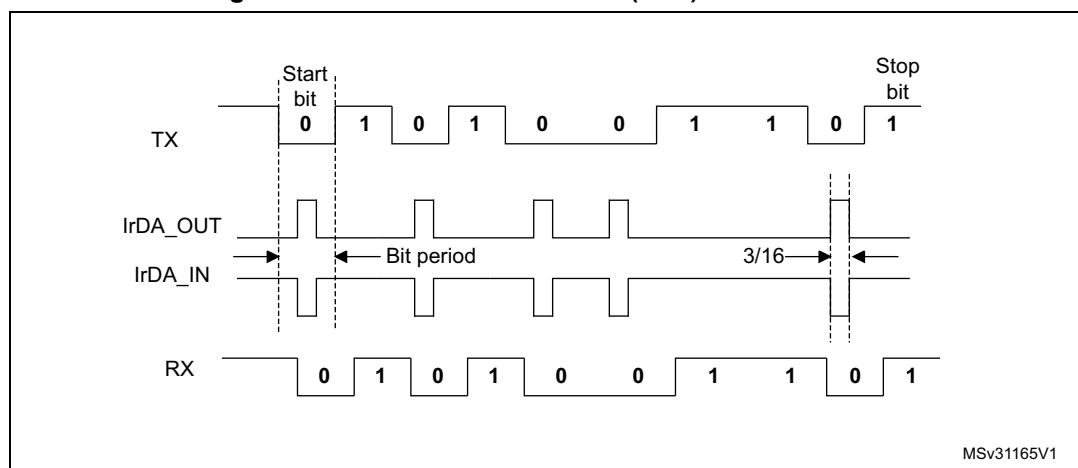
In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

- Receiver

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART\_GTPR).

**Note:** *A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.*

*The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).*

**Figure 259. IrDA SIR ENDEC block diagram****Figure 260. IrDA data modulation (3/16) - Normal mode**

### 24.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

**Note:** Refer to [Section 24.4: USART implementation on page 660](#) to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in [Section 24.5.6](#). To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART\_ISR register.

#### Transmission using DMA

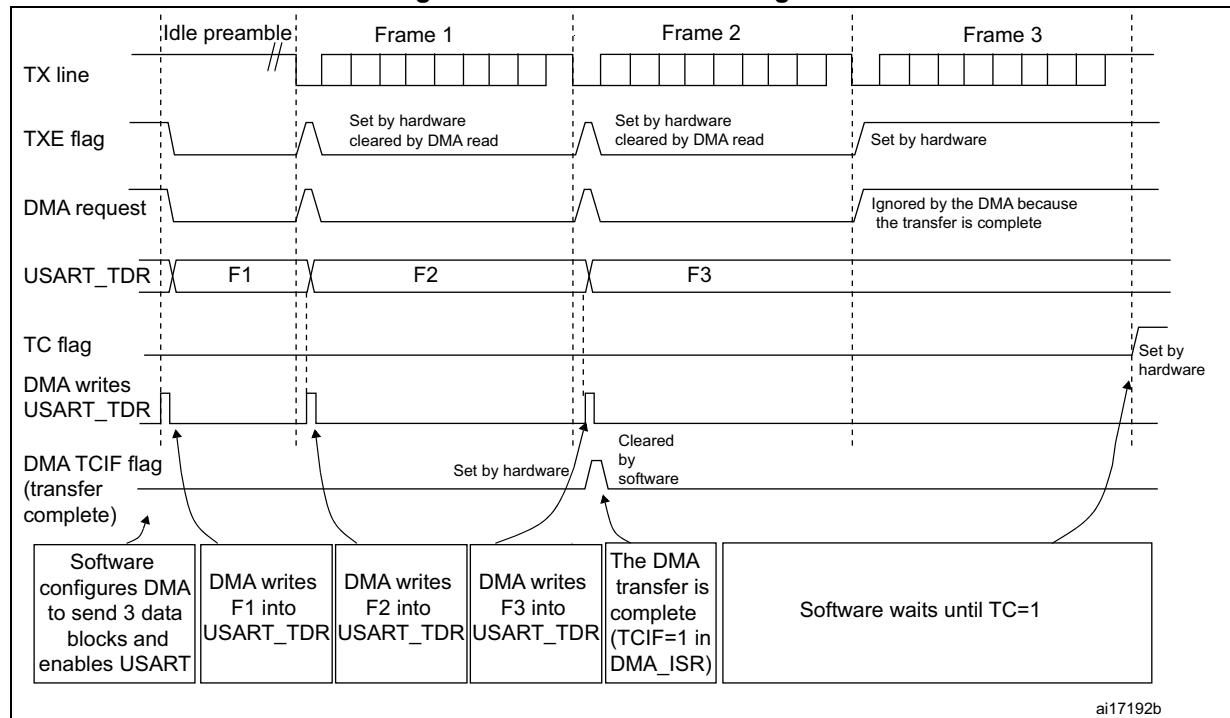
DMA mode can be enabled for transmission by setting DMAT bit in the USART\_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART\_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART\_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART\_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART\_ISR register by setting the TCCF bit in the USART\_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 261. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

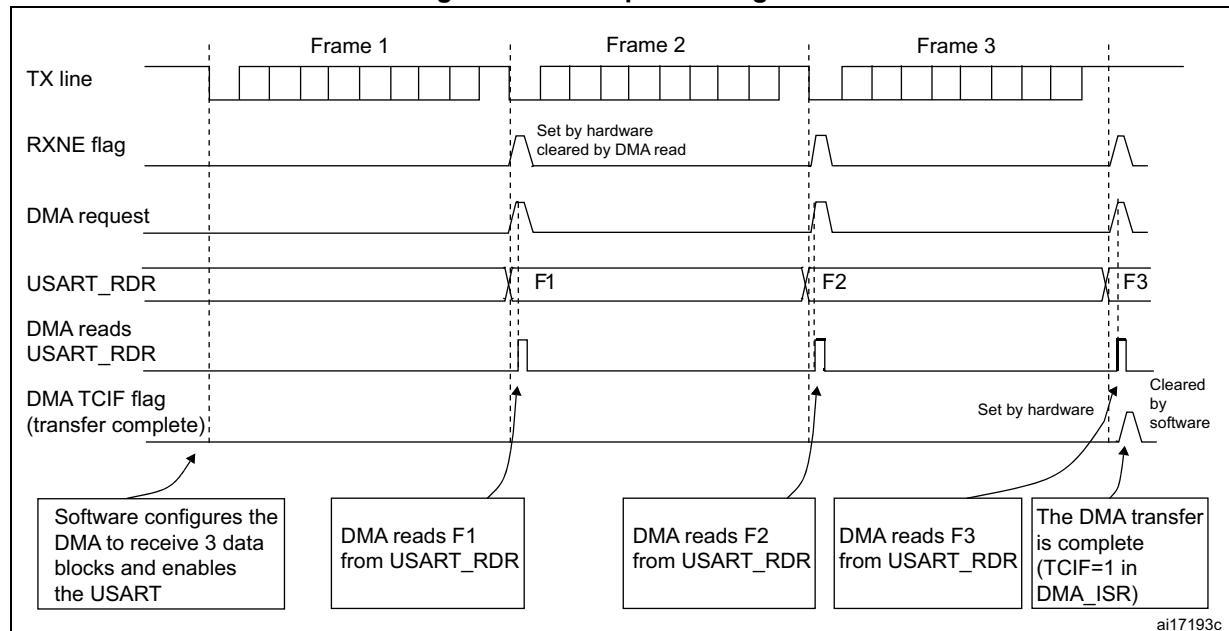
### Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART\_CR3 register. Data are loaded from the USART\_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART\_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART\_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 262. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

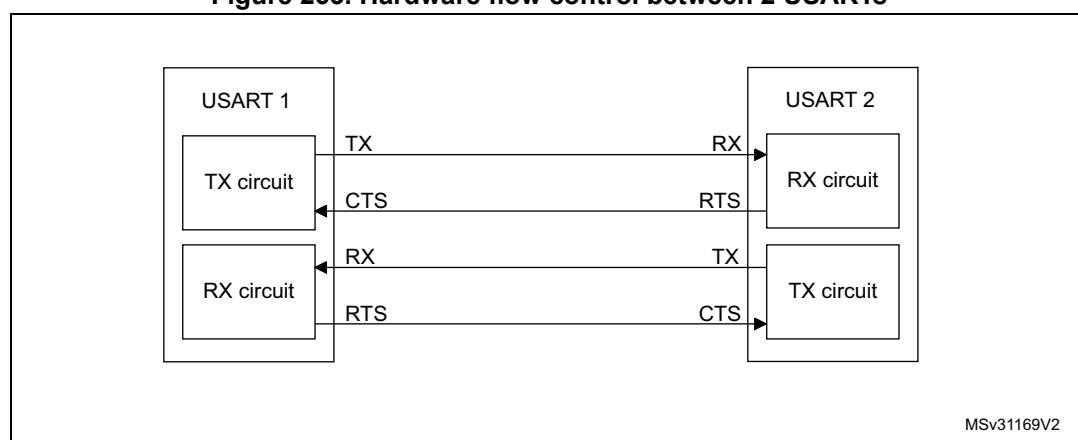
#### Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART\_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

#### 24.5.20 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The [Figure 263](#) shows how to connect 2 devices in this mode:

Figure 263. Hardware flow control between 2 USARTs

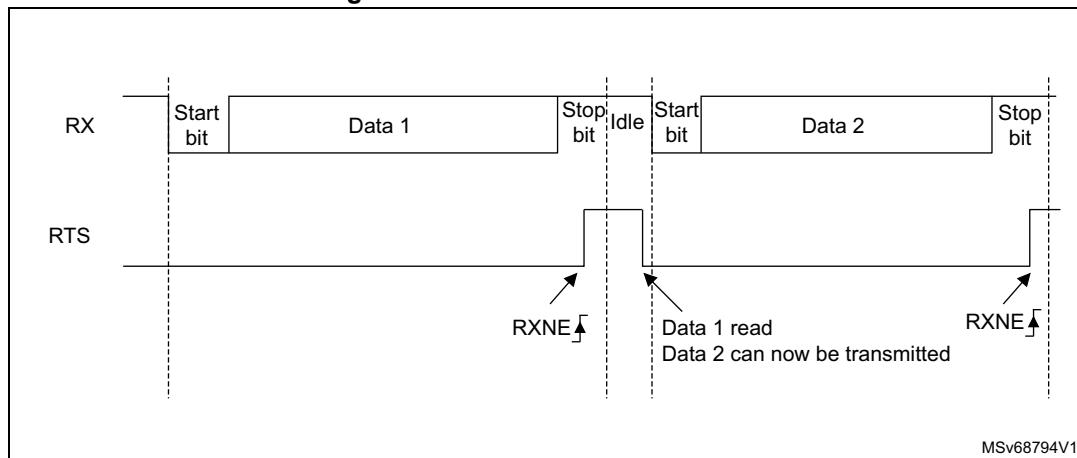


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to '1' in the USART\_CR3 register.

### RS232 RTS flow control

If the RTS flow control is enabled ( $RTSE = 1$ ), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 264](#) shows an example of communication with RTS flow control enabled.

**Figure 264. RS232 RTS flow control**



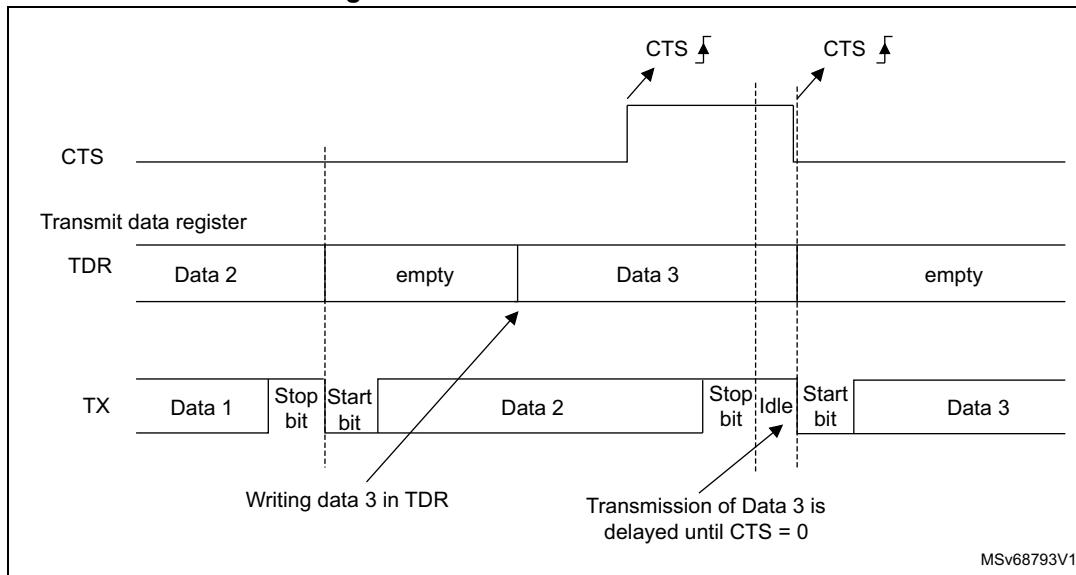
**Note:** When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

### RS232 CTS flow control

If the CTS flow control is enabled ( $CTSE = 1$ ), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if  $TXE/TXFE = 0$ ), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When  $CTSE = 1$ , the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART\_CR3 register is set. [Figure 265](#) shows an example of communication with CTS flow control enabled.

Figure 265. RS232 CTS flow control



**Note:** For correct behavior, CTS must be deasserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

### RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART\_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART\_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART\_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART\_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

### 24.5.21 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart\_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart\_pclk is gated, the USART provides a wake-up interrupt (**usart\_wkup**) if a specific action requiring the activation of the **usart\_pclk** clock is needed:

- If FIFO mode is disabled
    - usart\_pclk clock has to be activated to empty the USART data register.  
In this case, the usart\_wkup interrupt source is RXNE set to '1'. The RXNEIE bit must be set before entering low-power mode.
  - If FIFO mode is enabled
    - usart\_pclk clock has to be activated to:
      - to fill the TXFIFO
      - or to empty the RXFIFO
- In this case, the usart\_wkup interrupt source can be:
- RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart\_wkup interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart\_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart\_wkup** interrupt is generated if the WUFIE bit is set.

- Note: *Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.*
- The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.*
- When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.*
- When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.*
- When the FIFO is enabled, waking up from low-power mode on address match is only possible when Mute mode is enabled.*

### Using Mute mode with low-power mode

If the USART is put into Mute mode before entering low-power mode:

- Wake-up from Mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from Mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in Mute mode upon address match and wake up from low-power mode.

- Note: *When FIFO management is enabled, Mute mode can be used with wake-up from low-power mode without any constraints (i.e. the two points mentioned above about Mute and low-power mode are valid only when FIFO management is disabled).*

### Wake-up from low-power mode when USART kernel clock (usart\_ker\_ck) is OFF in low-power mode

If during low-power mode, the usart\_ker\_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart\_ker\_ck clock to be switched ON thanks to the usart\_ker\_ck\_req signal. usart\_ker\_ck is then used for the frame reception.

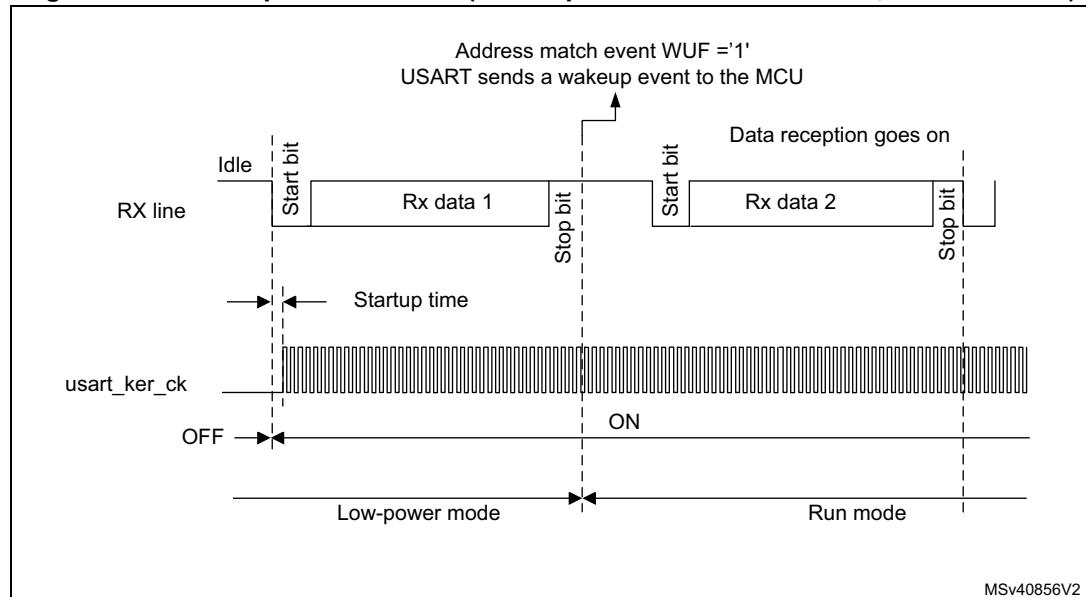
If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart\_ker\_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

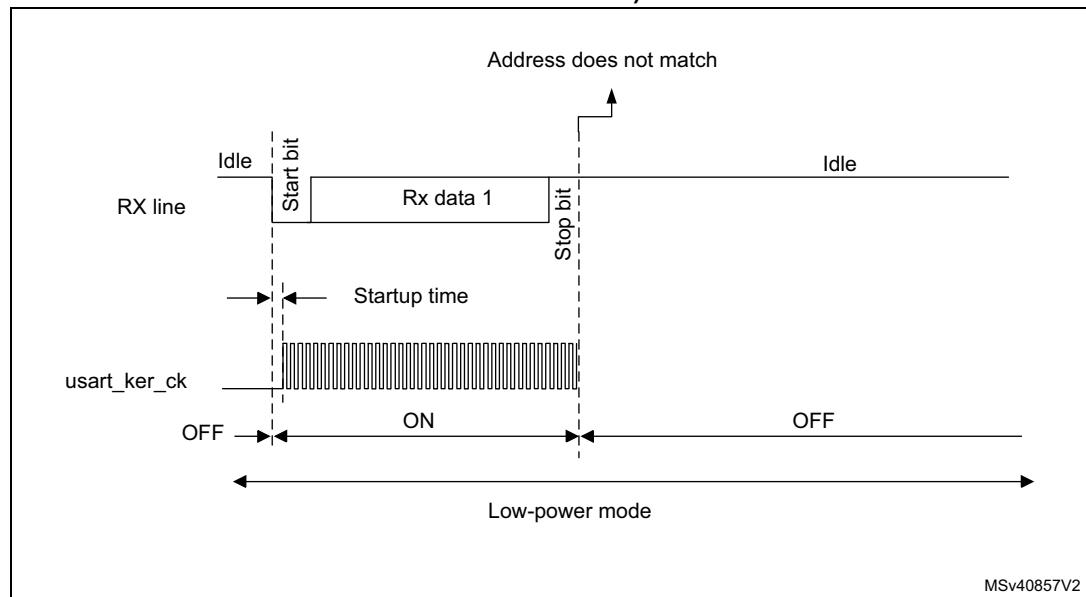
[Figure 266](#) shows the USART behavior when the wake-up event is verified.

**Figure 266. Wake-up event verified (wake-up event = address match, FIFO disabled)**



[Figure 267](#) shows the USART behavior when the wake-up event is not verified.

**Figure 267. Wake-up event not verified (wake-up event = address match, FIFO disabled)**



Note:

The figures above are valid when address match or any received frame is used as wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

### Determining the maximum USART baud rate that enables to correctly wake up the device from low-power mode

The maximum baud rate that enables to correctly wake up the device from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see [Section 24.5.8: Tolerance of the USART receiver to clock deviation](#)).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to [Table 108: Tolerance of the USART receiver when BRR \[3:0\] = 0000](#), the USART receiver tolerance equals 3.41%.

$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$

$$DWU_{\max} = t_{WUUSART} / (11 \times T_{bit\ Min})$$

$$T_{bit\ Min} = t_{WUUSART} / (11 \times DWU_{\max})$$

where  $t_{WUUSART}$  is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the `uart_ker_ck` inaccuracy.

For example, if HSI48 is used as `uart_ker_ck`, and the HSI48 inaccuracy is of 1%, then we obtain:

$t_{WUUSART} = 3\ \mu s$  (values provided only as examples; for correct values, refer to the device datasheet).

$$DWU_{\max} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit\ min} = 3\ \mu s / (11 \times 2.41\%) = 11.32\ \mu s$$

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is:  $1/11.32\ \mu s = 88.36\ Kbaud$ .

## 24.6 USART in low-power modes

Table 111. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop <sup>(1)</sup>	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

- Refer to [Section 24.4: USART implementation](#) to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

## 24.7 USART interrupts

Refer to [Table 112](#) for a detailed description of all USART interrupt requests.

**Table 112. USART interrupt requests**

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop <sup>(1)</sup> modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission Complete Before Guard Time	TCBGT	TCBGTE	Write TDR or write 1 in TCBGT		No	
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF <sup>(2)</sup>	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RXNEIE/RXFNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication	ORE <sup>(3)</sup>		Write 1 in ORECF		No	
	Framing Error in multibuffer communication	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

- The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to [Section 24.4: USART implementation](#) for the list of supported Stop modes.

2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART\_RDR. In Stop mode, USART\_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

## 24.8 USART registers

Refer to [Section 1.2 on page 35](#) for a list of abbreviations used in register descriptions.

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

### 24.8.1 USART control register 1 (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

#### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXF FIE	TXFEIE	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]								DEDT[4:0]	
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
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNE IE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when RXFF = 1 in the USART\_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when TXFE = 1 in the USART\_ISR register

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.*

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

*Note: In 7-bits data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bit 27 **EOBIE**: End-of-block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART\_ISR register

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. [Section 24.4: USART implementation on page 660](#).*

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

*Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.*

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART\_ISR register.

**Bit 13 MME:** Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

**Bit 12 M0:** Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

**Bit 11 WAKE:** Receiver wake-up method

This bit determines the USART wake-up method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 10 PCE:** Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 9 PS:** Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 8 PEIE:** PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART\_ISR register

**Bit 7 TXFNIE:** TXFIFO not-full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXFNF =1 in the USART\_ISR register

**Bit 6 TCIE:** Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART\_ISR register

**Bit 5 RXFNEIE:** RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART\_ISR register

**Bit 4 IDLEIE:** IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART\_ISR register

**Bit 3 TE:** Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART\_ISR register.*

*In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.*

**Bit 2 RE:** Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

**Bit 1 UESM:** USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

**Bit 0 UE:** USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART\_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

*In Smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.*

## 24.8.2 USART control register 1 [alternate] (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

### FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		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
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
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.

#### Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note:* FIFO mode can be used on standard UART communication, in SPI master/slave mode and in Smartcard modes only. It must not be enabled in IrDA and LIN modes.

#### Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

*Note:* In 7-bits data length mode, the Smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

#### Bit 27 **EOBIE**: End of Block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART\_ISR register

*Note:* If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).

#### Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART\_ISR register.

*Note:* If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. [Section 24.4: USART implementation on page 660](#).

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

*Note: In LIN, IrDA and Smartcard modes, this bit must be kept cleared.*

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART\_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART Mute mode function. When set, the USART can switch between active and Mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between Mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from Mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 9 PS:** Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 8 PEIE:** PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART\_ISR register

**Bit 7 TXEIE:** Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE = 1 in the USART\_ISR register

**Bit 6 TCIE:** Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART\_ISR register

**Bit 5 RXNEIE:** Receive data register not empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART\_ISR register

**Bit 4 IDLEIE:** IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART\_ISR register

**Bit 3 TE:** Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in Smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART\_ISR register.*

*In Smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.*

**Bit 2 RE:** Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

**Bit 1 UESM:** USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

**Bit 0 UE:** USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART\_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

*In Smartcard mode, (SCEN = 1), the CK pin is always available when CLKEN = 1, regardless of the UE bit value.*

### 24.8.3 USART 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[7:0]															
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
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

#### Bits 31:24 ADD[7:0]: Address of the USART node

These bits give the address of the USART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wake up from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

#### Bit 23 RTOEN: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART\_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

#### Bits 22:21 ABRMOD[1:0]: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 and Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

*Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)*

*If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

#### Bit 20 ABREN: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

#### Bit 19 MSBFIRST: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 18 DATAINV: Binary data inversion**

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 17 TXINV: TX pin active level inversion**

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd = 0/mark)

1: TX pin signal values are inverted ( $V_{DD} = 0/\text{mark}$ , Gnd = 1/idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 16 RXINV: RX pin active level inversion**

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , Gnd = 0/mark)

1: RX pin signal values are inverted ( $V_{DD} = 0/\text{mark}$ , Gnd = 1/idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 15 SWAP: Swap TX/RX pins**

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another USART.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 14 LINEN: LIN mode enable**

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART\_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value.*

*Refer to [Section 24.4: USART implementation on page 660](#).*

**Bits 13:12 STOP[1:0]: Stop bits**

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit can only be written when the USART is disabled (UE = 0).

*Note:* If neither synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).

In Smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:

UE = 0

SCEN = 1

GTPR configuration

CLKEN= 1

UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

*Note:* If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see [Figure 247](#) and [Figure 248](#))

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

*Note:* If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

**Caution:** The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART\_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

*Note:* If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).

## Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART\_ISR register

*Note:* If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).

**Bit 5 LBDL:** LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

**Bit 4 ADDM7:** 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

*Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.*

**Bit 3 DIS\_NSS:** NSS pin enable

When the DIS\_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

*Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

Bits 2:1 Reserved, must be kept at reset value.

**Bit 0 SLVEN:** Synchronous Slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.

1: Slave mode enabled.

*Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

*Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.*

#### 24.8.4 USART 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
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
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
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 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration

- 000:TXFIFO reaches 1/8 of its depth
- 001:TXFIFO reaches 1/4 of its depth
- 010:TXFIFO reaches 1/2 of its depth
- 011:TXFIFO reaches 3/4 of its depth
- 100:TXFIFO reaches 7/8 of its depth
- 101:TXFIFO becomes empty
- Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration

- 000:Receive FIFO reaches 1/8 of its depth
- 001:Receive FIFO reaches 1/4 of its depth
- 010:Receive FIFO reaches 1/2 of its depth
- 011:Receive FIFO reaches 3/4 of its depth
- 100:Receive FIFO reaches 7/8 of its depth
- 101:Receive FIFO becomes full
- Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission complete before guard time, interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever TCBGT=1 in the USART\_ISR register

*Note: If the USART does not support the Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated whenever WUF = 1 in the USART\_ISR register

*Note: WUFIE must be set before entering in low-power mode.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

- Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection  
 This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).  
 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)  
 01: Reserved.  
 10: WUF active on start bit detection  
 11: WUF active on RXNE/RXFNE.  
 This bitfield can only be written when the USART is disabled (UE = 0).  
*If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*
- Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count  
 This bitfield specifies the number of retries for transmission and reception in Smartcard mode.  
 In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).  
 In reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).  
 This bitfield must be programmed only when the USART is disabled (UE = 0).  
 When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.  
 0x0: retransmission disabled - No automatic retransmission in transmit mode.  
 0x1 to 0x7: number of automatic retransmission attempts (before signaling error)  
*Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **DEP**: Driver enable polarity selection  
 0: DE signal is active high.  
 1: DE signal is active low.  
 This bit can only be written when the USART is disabled (UE = 0).  
*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*
- Bit 14 **DEM**: Driver enable mode  
 This bit enables the user to activate the external transceiver control, through the DE signal.  
 0: DE function is disabled.  
 1: DE function is enabled. The DE signal is output on the RTS pin.  
 This bit can only be written when the USART is disabled (UE = 0).  
*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*
- Bit 13 **DDRE**: DMA Disable on reception error  
 0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred (used for Smartcard mode).  
 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE in case FIFO mode is enabled) before clearing the error flag.  
 This bit can only be written when the USART is disabled (UE=0).  
*Note: The reception errors are: parity error, framing error or noise error.*

Bit 12 **OVRDIS**: Overrun disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART\_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in USART\_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

*Note: This control bit enables checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **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 must be kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0).

If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

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

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

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling Smartcard mode.

0: Smartcard Mode disabled

1: Smartcard Mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 3 **HDSEL**: Half-duplex selection

Selection of Single-wire Half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART\_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART\_ISR register.

### 24.8.5 USART baud rate register (USART\_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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.
BRR[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 **BRR[15:0]**: USART baud rate

**BRR[15:4]**

BRR[15:4] = USARTDIV[15:4]

**BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

### 24.8.6 USART 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.
GT[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 bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If Smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bits 7:0 **PSC[7:0]**: Prescaler value

**In IrDA low-power and normal IrDA mode:**

PSC[7:0] = IrDA Normal and Low-Power baud rate

PSC[7:0] is used to program the prescaler for dividing the USART source clock to achieve the low-power frequency: the source clock is divided by the value given in the register (8 significant bits):

**In Smartcard mode:**

PSC[4:0] = Prescaler value

PSC[4:0] is used to program the prescaler for dividing the USART source clock to provide the Smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: Divides the source clock by 1 (IrDA mode) / by 2 (Smartcard mode)

00010: Divides the source clock by 2 (IrDA mode) / by 4 (Smartcard mode)

00011: Divides the source clock by 3 (IrDA mode) / by 6 (Smartcard mode)

...

11111: Divides the source clock by 31 (IrDA mode) / by 62 (Smartcard mode)

0010 0000: Divides the source clock by 32 (IrDA mode)

...

1111 1111: Divides the source clock by 255 (IrDA mode)

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: Bits [7:5] must be kept cleared if Smartcard mode is used.*

*This bitfield is reserved and forced by hardware to '0' when the Smartcard and IrDA modes are not supported. Refer to [Section 24.4: USART implementation on page 660](#).*

#### 24.8.7 USART 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]							
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
RTO[15:0]								RTO[15:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block length

This bitfield gives the Block length in Smartcard T = 1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0: 0 information characters + LEC

BLEN = 1: 0 information characters + CRC

BLEN = 255: 254 information characters + CRC (total 256 characters)

In Smartcard mode, the Block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

*Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.*

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In Smartcard mode, this value is used to implement the CWT and BWT. See Smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

*Note: This value must only be programmed once per received character.*

*Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.*

*This register is reserved and forced by hardware to “0x00000000” when the Receiver timeout feature is not supported. Refer to [Section 24.4: USART implementation on page 660](#).*

### 24.8.8 USART 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	w

Bits 31:5 Reserved, must be kept at reset value.

**Bit 4 TXFRQ:** Transmit data flush request

When FIFO mode is disabled, writing '1' to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in Smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART\_ISR register. If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART\_ISR register). Flushing the Transmit FIFO is supported in both UART and Smartcard modes.

*Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.*

**Bit 3 RXFRQ:** Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

**Bit 2 MMRQ:** Mute mode request

Writing 1 to this bit puts the USART in Mute mode and resets the RWU flag.

**Bit 1 SBKRQ:** Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

*Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.*

**Bit 0 ABRRQ:** Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART\_ISR and requests an automatic baud rate measurement on the next received data frame.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

## 24.8.9 USART interrupt and status register (USART\_ISR)

Address offset: 0x1C

Reset value: 0x0X80 00C0

X = 2 if FIFO/Smartcard mode is enabled

X = 0 if FIFO is enabled and Smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				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
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART\_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the USART\_CR3 register.

- 0: TXFIFO does not reach the programmed threshold.
- 1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in USART\_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART\_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the USART\_CR3 register.

- 0: Receive FIFO does not reach the programmed threshold.
- 1: Receive FIFO reached the programmed threshold.

*Note: When the RXFTCFG threshold is configured to '101', RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART\_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.*

Bit 25 **TCBGT**: Transmission complete before guard time flag

This bit is set when the last data written in the USART\_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART\_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART\_ICR register or by a write to the USART\_TDR register.

- 0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)
- 1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

*Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is '1'. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART\_RDR register).

An interrupt is generated if the RXFFIE bit = 1 in the USART\_CR1 register.

- 0: RXFIFO not full.
- 1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART\_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART\_CR1 register.

- 0: TXFIFO not empty.
- 1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART\_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART\_ICR register. An interrupt is generated if WUFIE = 1 in the USART\_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

Bit 19 **RWU**: Receiver wake-up from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART\_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART\_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 24.4: USART implementation on page 660.*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART\_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART\_ICR register.

An interrupt is generated if CMIE = 1 in the USART\_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF**: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case). It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed).

It is cleared by software, by writing 1 to the ABRRQ bit in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART\_TDR. This flag is reset by setting UDRCF bit in the USART\_ICR register.

0: No underrun error

1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

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 EOBIIE = 1 in the USART\_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART\_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART\_ICR register.

An interrupt is generated if RTOIE = 1 in the USART\_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.*

*The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.*

*If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART\_ICR register.

An interrupt is generated if CTSIE = 1 in the USART\_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **LBDIF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART\_ICR.

An interrupt is generated if LBDIE = 1 in the USART\_CR2 register.

0: LIN Break not detected

1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value.  
Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 7 **TXFNF**: TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART\_TDR. Every write operation to the USART\_TDR places the data in the TXFIFO.

This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART\_TDR.

An interrupt is generated if the TXFNIE bit =1 in the USART\_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

*Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).*

*This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART\_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

An interrupt is generated if TCIE = 1 in the USART\_CR1 register.

TC bit is cleared by software, by writing 1 to the TCCF in the USART\_ICR register or by a write to the USART\_TDR register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.*

**Bit 5 RXFNE:** RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART\_RDR register. Every read operation from the USART\_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART\_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

**Bit 4 IDLE:** Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).*

*If Mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

**Bit 3 ORE:** Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART\_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART\_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART\_CR1 register, or EIE = 1 in the USART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the USART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART\_CR3 register.*

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART\_ICR register.

0: No noise is detected

1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to [Section 24.5.8: Tolerance of the USART receiver to clock deviation on page 678](#)).*

*This error is associated with the character in the USART\_RDR.*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART\_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART\_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

*Note: This error is associated with the character in the USART\_RDR.*

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECE bit in the USART\_ICR register.

An interrupt is generated if PEIE = 1 in the USART\_CR1 register.

0: No parity error

1: Parity error

*Note: This error is associated with the character in the USART\_RDR.*

**24.8.10 USART interrupt and status register [alternate] (USART\_ISR)**

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			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	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT**: Transmission complete before guard time flag

This bit is set when the last data written in the USART\_TDR has been transmitted correctly out of the shift register.

It is set by hardware in Smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART\_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART\_ICR register or by a write to the USART\_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

*Note: If the USART does not support the Smartcard mode, this bit is reserved and kept at reset value. If the USART supports the Smartcard mode and the Smartcard mode is enabled, the TCBGT reset value is '1'. Refer to [Section 24.4: USART implementation on page 660](#).*

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART\_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART\_ICR register. An interrupt is generated if WUFIE = 1 in the USART\_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 19 **RWU**: Receiver wake-up from Mute mode

This bit indicates if the USART is in Mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The Mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART\_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART\_RQR register.

0: Receiver in active mode

1: Receiver in Mute mode

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART\_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART\_ICR register.

An interrupt is generated if CMIE = 1 in the USART\_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF**: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART\_TDR. This flag is reset by setting UDRCF bit in the USART\_ICR register.

0: No underrun error

1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

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 EOBIIE = 1 in the USART\_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART\_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

*Note: If Smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART\_ICR register.

An interrupt is generated if RTOIE = 1 in the USART\_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.*

*The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.*

*If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART\_ICR register.

An interrupt is generated if CTSIE = 1 in the USART\_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART\_ICR.

An interrupt is generated if LBDIE = 1 in the USART\_CR2 register.

0: LIN Break not detected

1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value.  
Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 7 **TXE**: Transmit data register empty

TXE is set by hardware when the content of the USART\_TDR register has been transferred into the shift register. It is cleared by writing to the USART\_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART\_RQR register, in order to discard the data (only in Smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXIE bit = 1 in the USART\_CR1 register.

0: Data register full

1: Data register not full

**Bit 6 TC:** Transmission complete

This bit indicates that the last data written in the USART\_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART\_CR1 register.

TC bit is cleared by software, by writing 1 to the TCCF in the USART\_ICR register or by a write to the USART\_TDR register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.*

**Bit 5 RXNE:** Read data register not empty

RXNE bit is set by hardware when the content of the USART\_RDR shift register has been transferred to the USART\_RDR register. It is cleared by reading from the USART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART\_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

**Bit 4 IDLE:** Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).*

*If Mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the Mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

**Bit 3 ORE:** Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART\_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART\_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART\_CR1 register, or EIE = 1 in the USART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the USART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART\_CR3 register.*

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART\_ICR register.

- 0: No noise is detected
- 1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to [Section 24.5.8: Tolerance of the USART receiver to clock deviation on page 678](#)).*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART\_ICR register. When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART\_CR3 register.

- 0: No Framing error is detected
- 1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART\_ICR register.

An interrupt is generated if PEIE = 1 in the USART\_CR1 register.

- 0: No parity error
- 1: Parity error

**24.8.11 USART 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.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGT CF	TCCF	TXFEC F	IDLEC F	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART\_ISR register.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART\_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART\_ISR register.

*Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#)*

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART\_ISR register.

*Note: If the USART does not support Smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the USART\_ISR register.

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 8 **LBDCF**: LIN break detection clear flag

Writing 1 to this bit clears the LBDF flag in the USART\_ISR register.

*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 24.4: USART implementation on page 660](#).*

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag

Writing 1 to this bit clears the TCBGT flag in the USART\_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the USART\_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag

Writing 1 to this bit clears the TXFE flag in the USART\_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the USART\_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the USART\_ISR register.

Bit 2 **NECF**: Noise detected clear flag

Writing 1 to this bit clears the NE flag in the USART\_ISR register.

Bit 1 **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the USART\_ISR register.

Bit 0 **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the USART\_ISR register.

### 24.8.12 USART receive data register (USART\_RDR)

Address offset: 0x24

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.	RDR[8:0]														
								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 241](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 24.8.13 USART transmit data register (USART\_TDR)

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.								
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

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

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART\_TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 241](#)).

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/TXFNF = 1.

### 24.8.14 USART prescaler register (USART\_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	rw	rw	rw	rw
PRESCALER[3:0]															

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Remaining combinations: Reserved

*Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.*

### 24.8.15 USART register map

The table below gives the USART register map and reset values.

**Table 113. USART register map and reset values**

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5
0x00	USART_CR1 FIFO enabled	RXFFIE	0	TXFEIE	0																							
	Reset value	0	Res.	0	Res.																							
0x00	USART_CR1 FIFO disabled			FIFOEN	0																							
	Reset value		0	0	0	M1	0	M1																				
0x04	USART_CR2																											
	Reset value	0	0	0	TXFTCFG[2:0]	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																											
	Reset value	0	0	0	TXFTCFG[2:0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	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																											
0x10	USART_GTPR	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																											
0x14	USART_RTOR																											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	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.
	Reset value																											
0x1C	USART_ISR FIFO mode enabled	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 FIFO mode disabled	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	USART_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.
	Reset value																											
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.	Res.	Res.	Res.	Res.
	Reset value																											

**Table 113. USART register map and reset values (continued)**

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	USART_TDR	Res.																															
	Reset value	Res.																															
0x2C	USART_PRESC	Res.																															
	Reset value	Res.																															

Refer to [Section 2.2: Memory organization](#) for the register boundary addresses.

## 25 Serial peripheral interface / integrated interchip sound (SPI/I2S)

### 25.1 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 integrated interchip sound (I<sup>2</sup>S) protocol is also a synchronous serial communication interface. It can operate in slave or master mode with half-duplex communication. It can address four different audio standards including the Philips I<sup>2</sup>S standard, the MSB- and LSB-justified standards and the PCM standard.

### 25.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to  $f_{PCLK}/2$
- Slave mode frequency up to  $f_{PCLK}/2$ .
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support

## 25.3 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. The ratio is fixed at  $256 \times f_s$  for all I2S modes, and to  $128 \times f_s$  for all PCM modes (where  $f_s$  is the audio sampling frequency).

## 25.4 SPI/I2S implementation

The following table describes all the SPI instances and their features embedded in the devices.

**Table 114. STM32C0x1 SPI and SPI/I2S implementation**

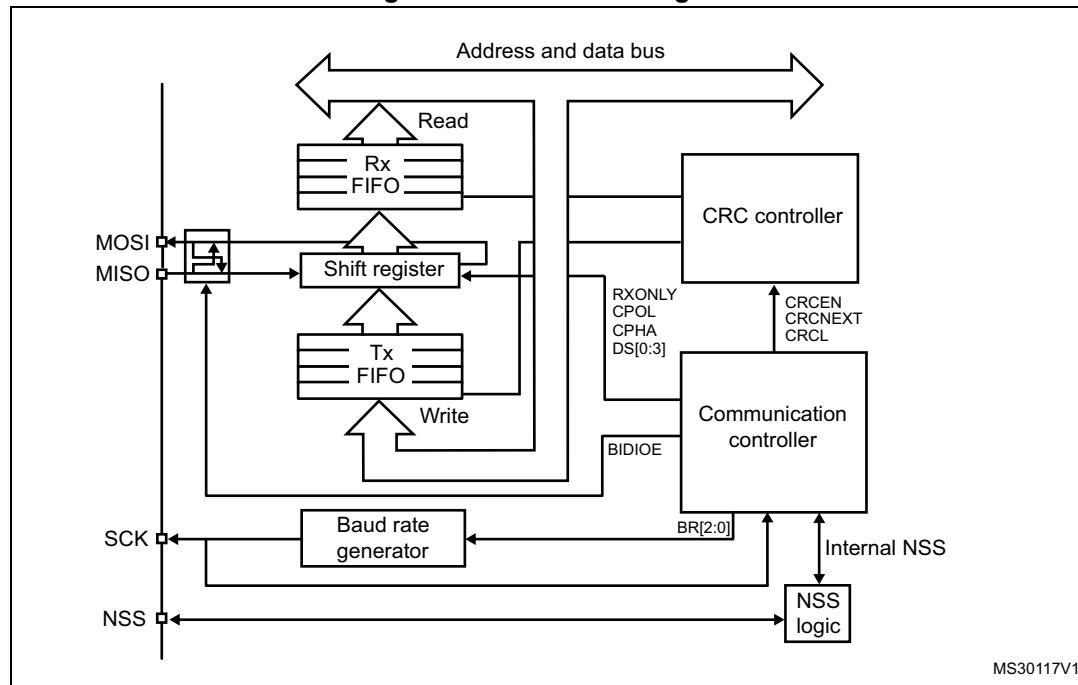
SPI Features	SPI1 / I2S1
Enhanced NSSP & TI modes	Yes
I2S support	Yes
Hardware CRC calculation	Yes
Data size configuration	from 4 to 16-bit
Rx/Tx FIFO size	32-bit
Wakeup capability from Low-power Sleep	Yes

## 25.5 SPI functional description

### 25.5.1 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 268](#).

**Figure 268. 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 25.5.5: Slave select \(NSS\) pin management](#) for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires - one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management.

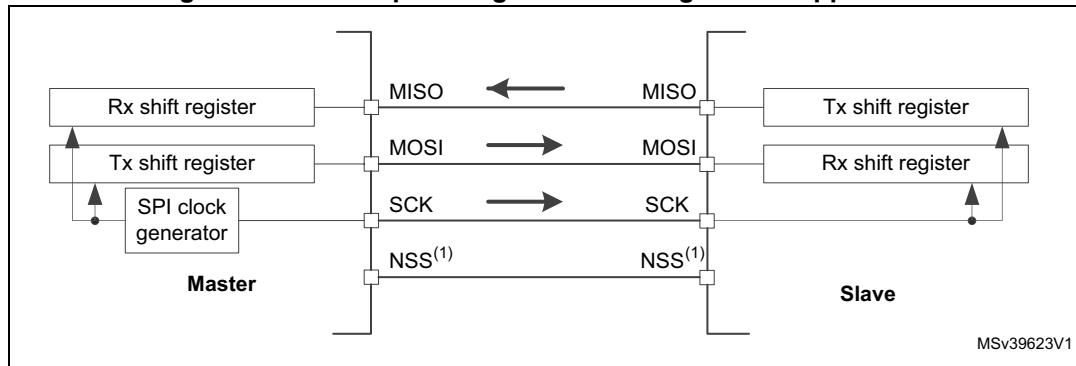
## 25.5.2 Communications between one master and one slave

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 269. Full-duplex single master/ single slave application**

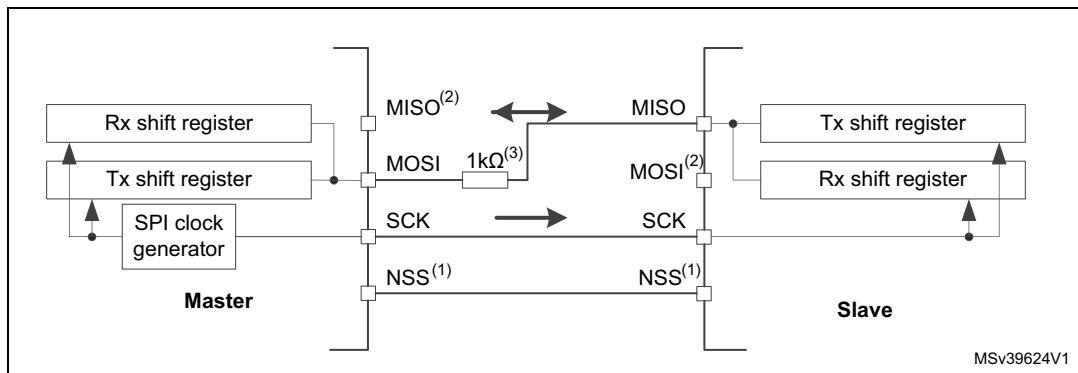


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 25.5.5: Slave select \(NSS\) pin management](#).

### Half-duplex communication

The SPI can communicate in half-duplex mode by setting the BIDIMODE bit in the SPIx\_CR1 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data is synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the BDIOE bit in their SPIx\_CR1 registers. In this configuration, the master's MISO pin and the slave's MOSI pin are free for other application uses and act as GPIOs.

Figure 270. Half-duplex single master/ single slave application



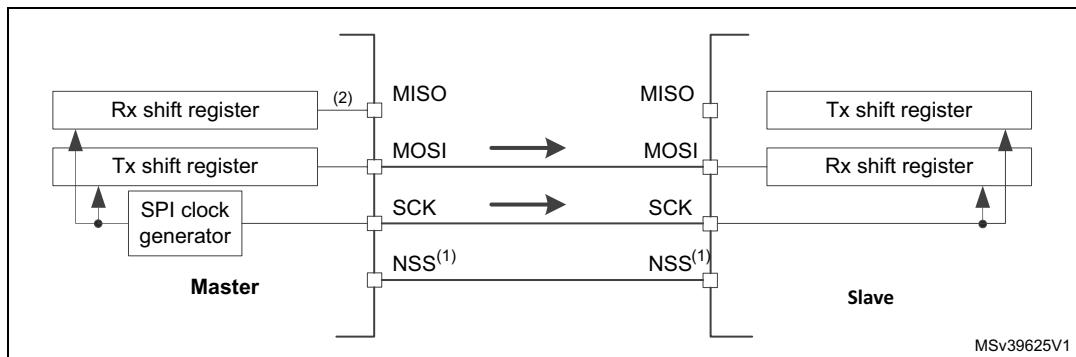
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 25.5.5: Slave select \(NSS\) pin management](#).
2. In this configuration, the master's MISO pin and the slave's MOSI pin can be used as GPIOs.
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectional mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communication data). Both nodes then fight while providing opposite output levels on the common line temporary till next node changes its direction settings correspondingly, too. It is suggested to insert a serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation.

### Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the RXONLY bit in the SPIx\_CR1 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO and MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode (RXONLY=0):** The configuration settings are the same as for full-duplex. The application has to ignore the information captured on the unused input pin. This pin can be used as a standard GPIO.
- **Receive-only mode (RXONLY=1):** The application can disable the SPI output function by setting the RXONLY bit. In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see [25.5.5: Slave select \(NSS\) pin management](#)). Received data events appear depending on the data buffer configuration. In the master configuration, the MOSI output is disabled and the pin can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled. The only way to stop the clock is to clear the RXONLY bit or the SPE bit and wait until the incoming pattern from the MISO pin is finished and fills the data buffer structure, depending on its configuration.

**Figure 271. Simplex single master/single slave application (master in transmit-only/slave in receive-only mode)**



1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 25.5.5: Slave select \(NSS\) pin management](#).
2. An accidental input information is captured at the input of transmitter Rx shift register. All the events associated with the transmitter receive flow must be ignored in standard transmit only mode (e.g. OVR flag).
3. In this configuration, both the MISO pins can be used as GPIOs.

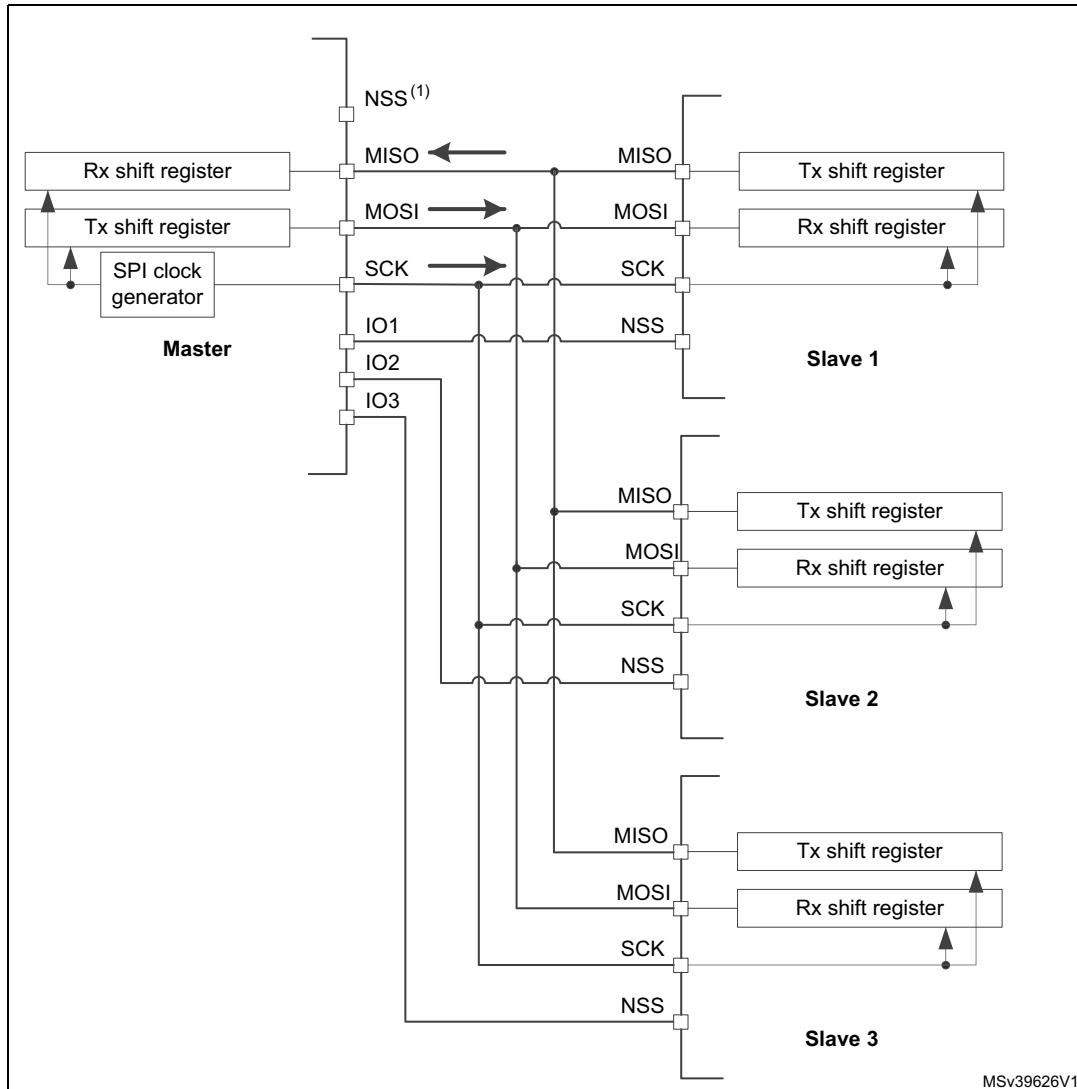
**Note:**

*Any simplex communication can be alternatively replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled while BDIO bit is not changed).*

### 25.5.3 Standard multislide 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 272](#)). 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 272. Master and three independent slaves



1. NSS pin is not used on master side at this configuration. It has to be managed internally ( $SSM=1$ ,  $SSI=1$ ) to prevent any MODF error.
2. As MISO pins of the slaves are connected together, all slaves must have the GPIO configuration of their MISO pin set as alternate function open-drain (see I/O alternate function input/output section (GPIO)).

#### 25.5.4 Multimaster communication

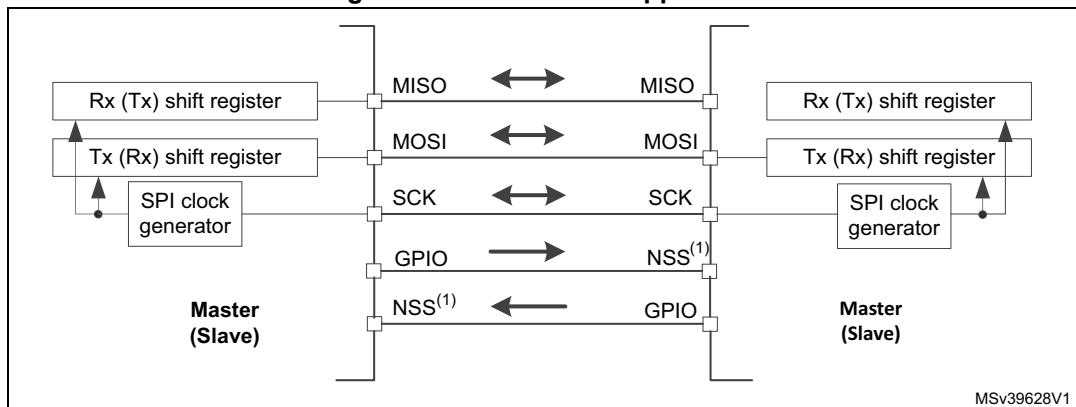
Unless SPI bus is not designed for a multimaster capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

**Figure 273. Multimaster application**



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

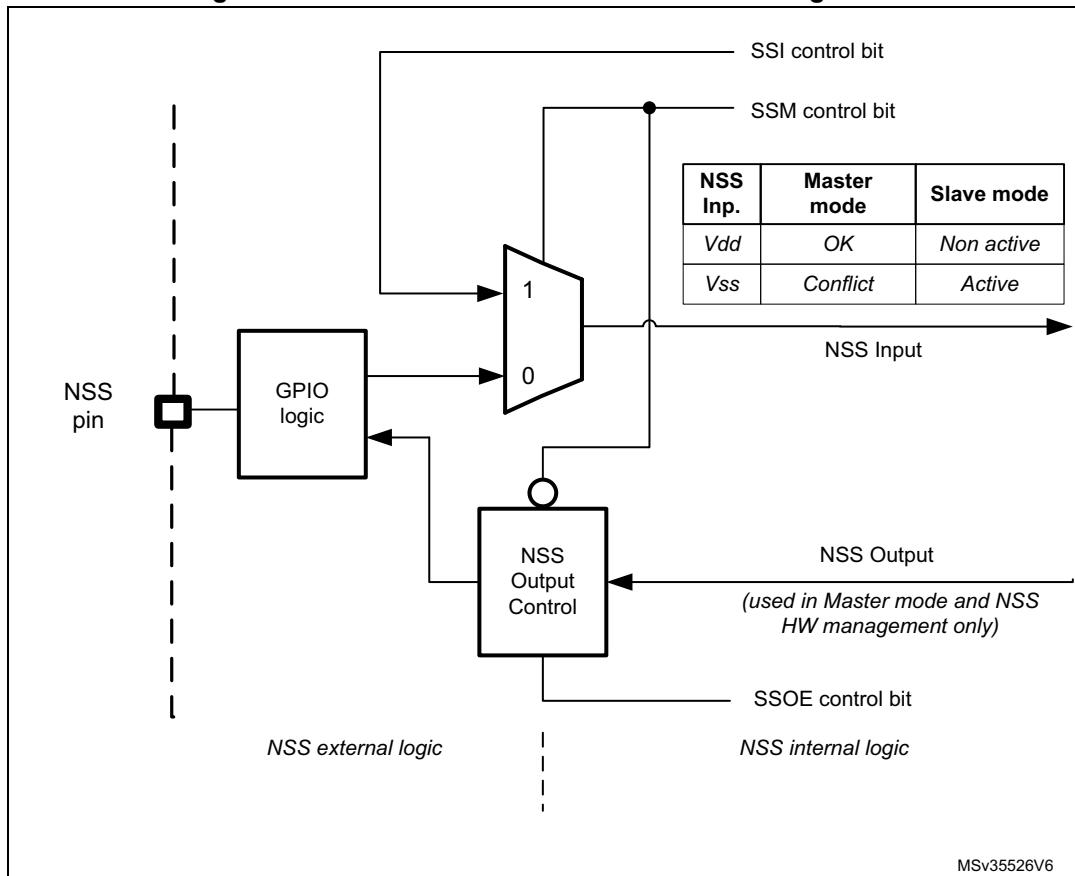
### 25.5.5 Slave select (NSS) pin management

In slave mode, the NSS works as a standard “chip select” input and lets the slave communicate with the master. In master mode, NSS can be used either as output or input. As an input it can prevent multimaster bus collision, and as an output it can drive a slave select signal of a single slave.

Hardware or software slave select management can be set using the SSM bit in the SPIx\_CR1 register:

- **Software NSS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in register SPIx\_CR1. The external NSS pin is free for other application uses.
- **Hardware NSS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in register SPIx\_CR1).
  - **NSS output enable (SSM=0,SSOE = 1):** this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode (SPE=1), and is kept low until the SPI is disabled (SPE =0). A pulse can be generated between continuous communications if NSS pulse mode is activated (NSSP=1). The SPI cannot work in multimaster configuration with this NSS setting.
  - **NSS output disable (SSM=0, SSOE = 0):** if the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled low in this mode, the SPI enters master mode fault state and the device is automatically reconfigured in slave mode. In slave mode, the NSS pin works as a standard “chip select” input and the slave is selected while NSS line is at low level.

Figure 274. Hardware/software slave select management



## 25.5.6 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slaves devices must follow the same communication format.

### Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx\_CR1 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

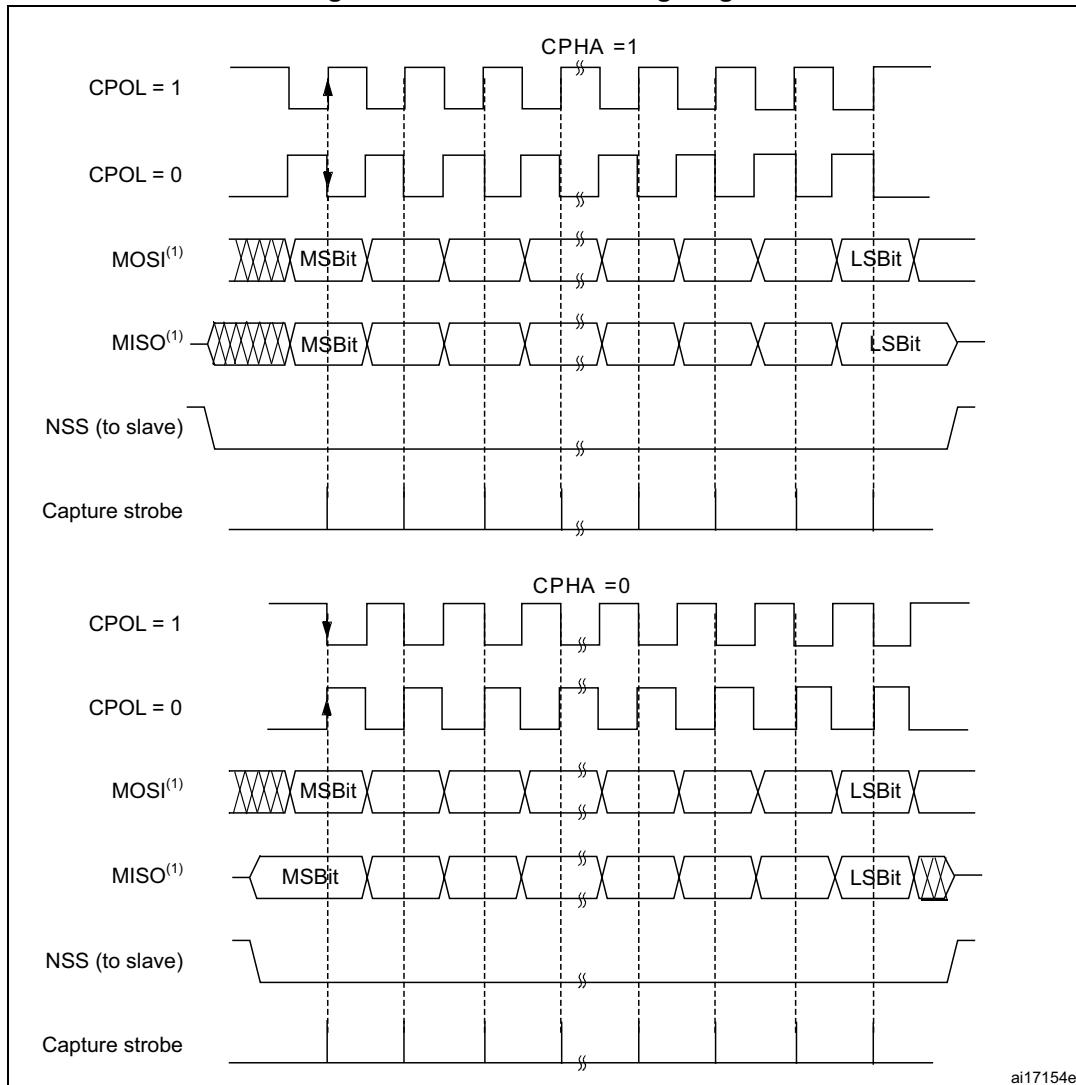
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

[Figure 275](#), 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).

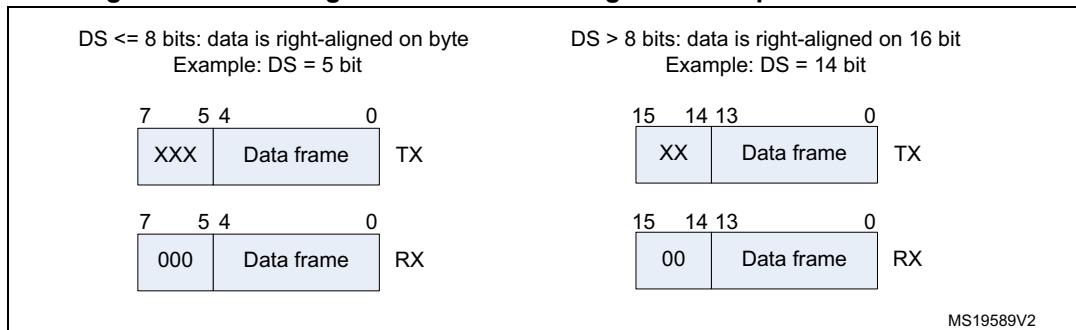
**Figure 275. Data clock timing diagram**



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 276](#)). During communication, only bits within the data frame are clocked and transferred.

**Figure 276. 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.

### 25.5.7 Configuration of SPI

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write proper GPIO registers: Configure GPIO for MOSI, MISO and SCK pins.
2. Write to the SPI\_CR1 register:
  - a) Configure the serial clock baud rate using the BR[2:0] bits (Note: 4).
  - b) Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock (CPHA must be cleared in NSSP mode). (Note: 2 - except the case when CRC is enabled at TI mode).
  - c) Select simplex or half-duplex mode by configuring RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE cannot be set at the same time).
  - d) Configure the LSBFIRST bit to define the frame format (Note: 2).
  - e) Configure the CRCL and CRCEN bits if CRC is needed (while SCK clock signal is at idle state).
  - f) Configure SSM and SSI (Notes: 2 & 3).
  - g) Configure the MSTR bit (in multimaster NSS configuration, avoid conflict state on NSS if master is configured to prevent MODF error).
3. Write to SPI\_CR2 register:
  - a) Configure the DS[3:0] bits to select the data length for the transfer.
  - b) Configure SSOE (Notes: 1 & 2 & 3).
  - c) Set the FRF bit if the TI protocol is required (keep NSSP bit cleared in TI mode).
  - d) Set the NSSP bit if the NSS pulse mode between two data units is required (keep CHPA and TI bits cleared in NSSP mode).
  - e) Configure the RXTH 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

### 25.5.8 Procedure for enabling SPI

It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave must already contain data to be sent before starting communication with the master (either on the first edge of the communication clock, or before the end of the ongoing communication if the clock signal is continuous). The SCK signal must be settled at an idle state level corresponding to the selected polarity before the SPI slave is enabled.

The master at full-duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled and TXFIFO is not empty, or with the next write to TXFIFO.

In any master receive only mode (RXONLY=1 or BIDIMODE=1 & BIDIOE=0), master starts to communicate and the clock starts running immediately after SPI is enabled.

For handling DMA, follow the dedicated section.

### 25.5.9 Data transmission and reception procedures

#### RXFIFO and TXFIFO

All SPI data transactions pass through the 32-bit embedded FIFOs. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each direction has its own FIFO called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes except for receiver-only mode (slave or master) with CRC calculation enabled (see [Section 25.5.14: CRC calculation](#)).

The handling of FIFOs depends on the data exchange mode (duplex, simplex), data frame format (number of bits in the frame), access size performed on the FIFO data registers (8-bit or 16-bit), and whether or not data packing is used when accessing the FIFOs (see [Section 25.5.13: TI mode](#)).

A read access to the SPIx\_DR register returns the oldest value stored in RXFIFO that has not been read yet. A write access to the SPIx\_DR stores the written data in the TXFIFO at the end of a send queue. The read access must be always aligned with the RXFIFO threshold configured by the FRXTH bit in SPIx\_CR2 register. FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy level of both FIFOs.

A read access to the SPIx\_DR register must be managed by the RXNE event. This event is triggered when data is stored in RXFIFO and the threshold (defined by FRXTH bit) is reached. When RXNE is cleared, RXFIFO is considered to be empty. In a similar way, write access of a data frame to be transmitted is managed by the TXE event. This event is triggered when the TXFIFO level is less than or equal to half of its capacity. Otherwise TXE is cleared and the TXFIFO is considered as full. In this way, 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 278](#) through [Figure 281](#).

Another way to manage the data exchange is to use DMA (see [Communication using DMA \(direct memory addressing\)](#)).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at [Section 25.5.10: SPI status flags](#)). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

### Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half-duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislide 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 25.5.5: Slave select \(NSS\) pin management](#)).

When the BSY bit is set it signifies an ongoing data frame transaction. When the dedicated frame transaction is finished, the RXNE flag is raised. The last bit is just sampled and the complete data frame is stored in the RXFIFO.

### Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph. It is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Ongoing transactions can be corrupted in this case. In some modes the disable procedure is the only way to stop continuous communication running.

Master in full-duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. Special care must be taken in packing mode when an odd number of data frames are transacted to prevent some dummy byte exchange (refer to [Data packing](#) section). Before the SPI is disabled in these modes, the user must follow standard disable procedure. When

the SPI is disabled at the master transmitter while a frame transaction is ongoing or next data frame is stored in TXFIFO, the SPI behavior is not guaranteed.

When the master is in any receive only mode, the only way to stop the continuous clock is to disable the peripheral by SPE=0. This must occur in specific time window within last data frame transaction just between the sampling time of its first bit and before its last bit transfer starts (in order to receive a complete number of expected data frames and to prevent any additional “dummy” data reading after the last valid data frame). Specific procedure must be followed when disabling SPI in this mode.

Data received but not read remains stored in RXFIFO when the SPI is disabled, and must be processed the next time the SPI is enabled, before starting a new sequence. To prevent having unread data, ensure that RXFIFO is empty when disabling the SPI, by using the correct disabling procedure, or by initializing all the SPI registers with a software reset via the control of a specific register dedicated to peripheral reset (see the SPIiRST bits in the RCC\_APBiRSTR registers).

Standard disable procedure is based on pulling BSY status together with FTLVL[1:0] to check if a transmission session is fully completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When NSS signal is managed by software and master has to provide proper end of NSS pulse for slave, or
- When transactions’ streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

The correct disable procedure is (except when receive only mode is used):

1. Wait until FTLVL[1:0] = 00 (no more data to transmit).
2. Wait until BSY=0 (the last data frame is processed).
3. Disable the SPI (SPE=0).
4. Read data until FRLVL[1:0] = 00 (read all the received data).

The correct disable procedure for certain receive only modes is:

1. Interrupt the receive flow by disabling SPI (SPE=0) in the specific time window while the last data frame is ongoing.
2. Wait until BSY=0 (the last data frame is processed).
3. Read data until FRLVL[1:0] = 00 (read all the received data).

**Note:**

*If packing mode is used and an odd number of data frames with a format less than or equal to 8 bits (fitting into one byte) has to be received, FRXTH must be set when FRLVL[1:0] = 01, in order to generate the RXNE event to read the last odd data frame and to keep good FIFO pointer alignment.*

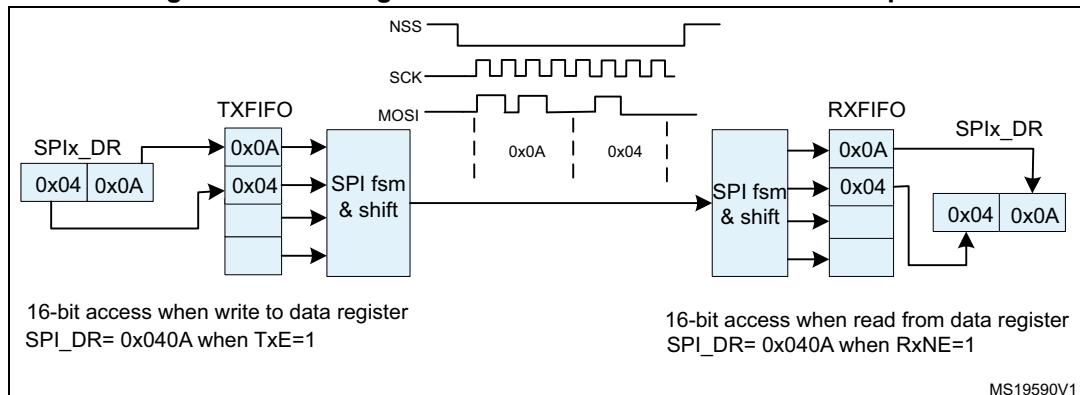
### Data packing

When the data frame size fits into one byte (less than or equal to 8 bits), data packing is used automatically when any read or write 16-bit access is performed on the SPIx\_DR register. The double data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other half stored in the MSB. [Figure 277](#) 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 277. Packing data in FIFO for transmission and reception**



1. In this example: Data size DS[3:0] is 4-bit configured, CPOL=0, CPHA=1 and LSBFIRST =0. The Data storage is always right aligned while the valid bits are performed on the bus only, the content of LSB byte goes first on the bus, the unused bits are not taken into account on the transmitter side and padded by zeros at the receiver side.

### Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bit in the SPIx\_CR2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPIx\_DR register.
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPIx\_DR register.

See [Figure 278](#) through [Figure 281](#).

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 759](#).)

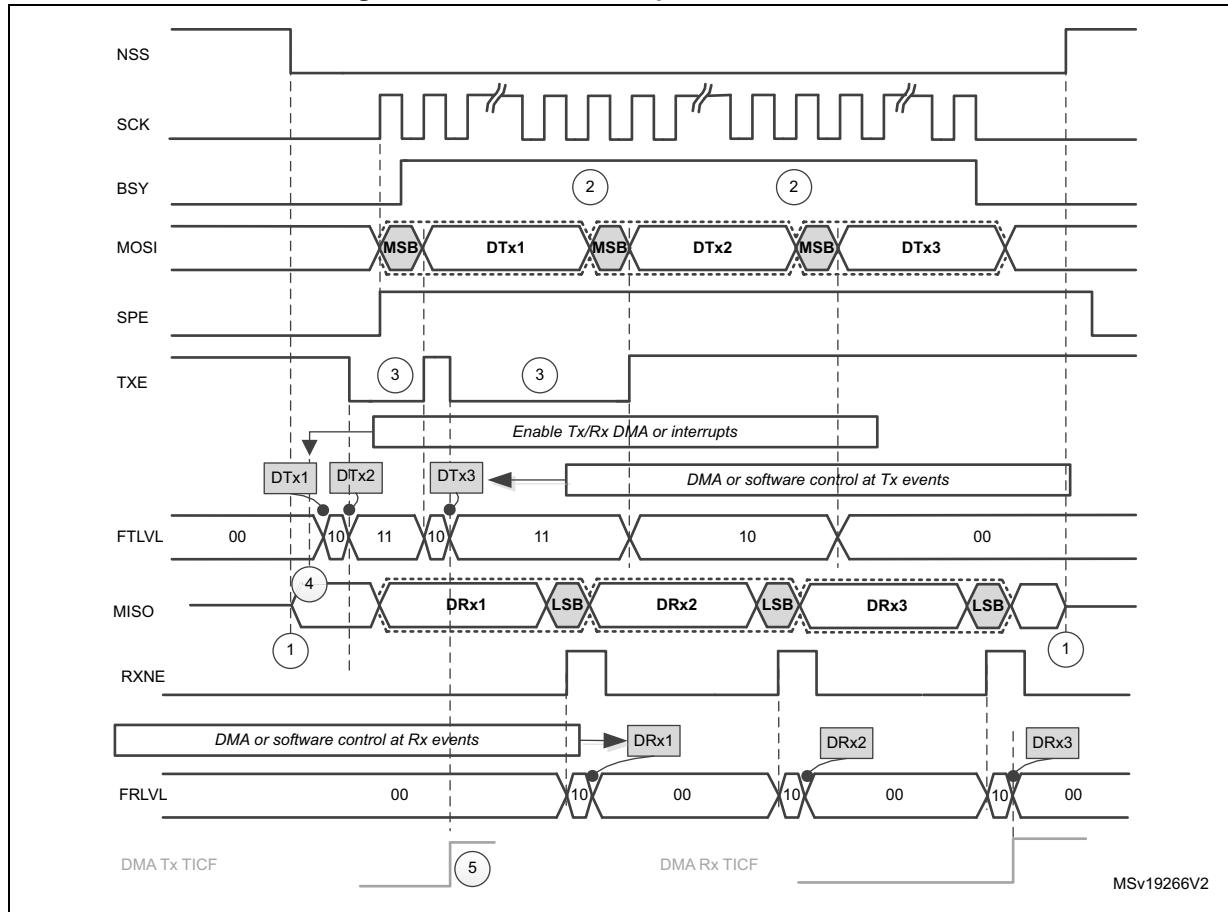
## Communication diagrams

Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for [Figure 278 on page 763](#) through [Figure 281 on page 766](#):

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 278. Master full-duplex communication



Assumptions for master full-duplex communication example:

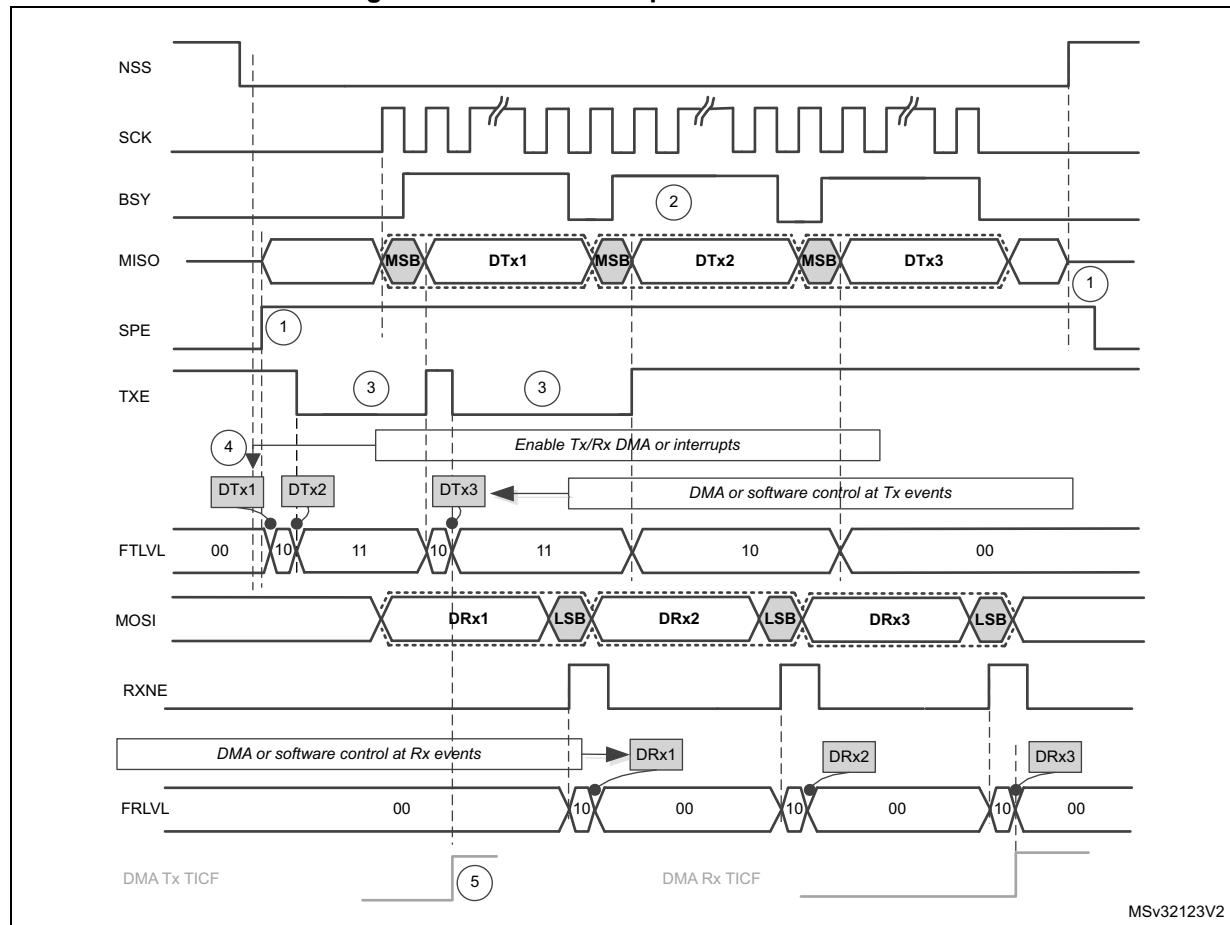
- Data size > 8 bit

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 762](#) for details about common assumptions and notes.

Figure 279. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

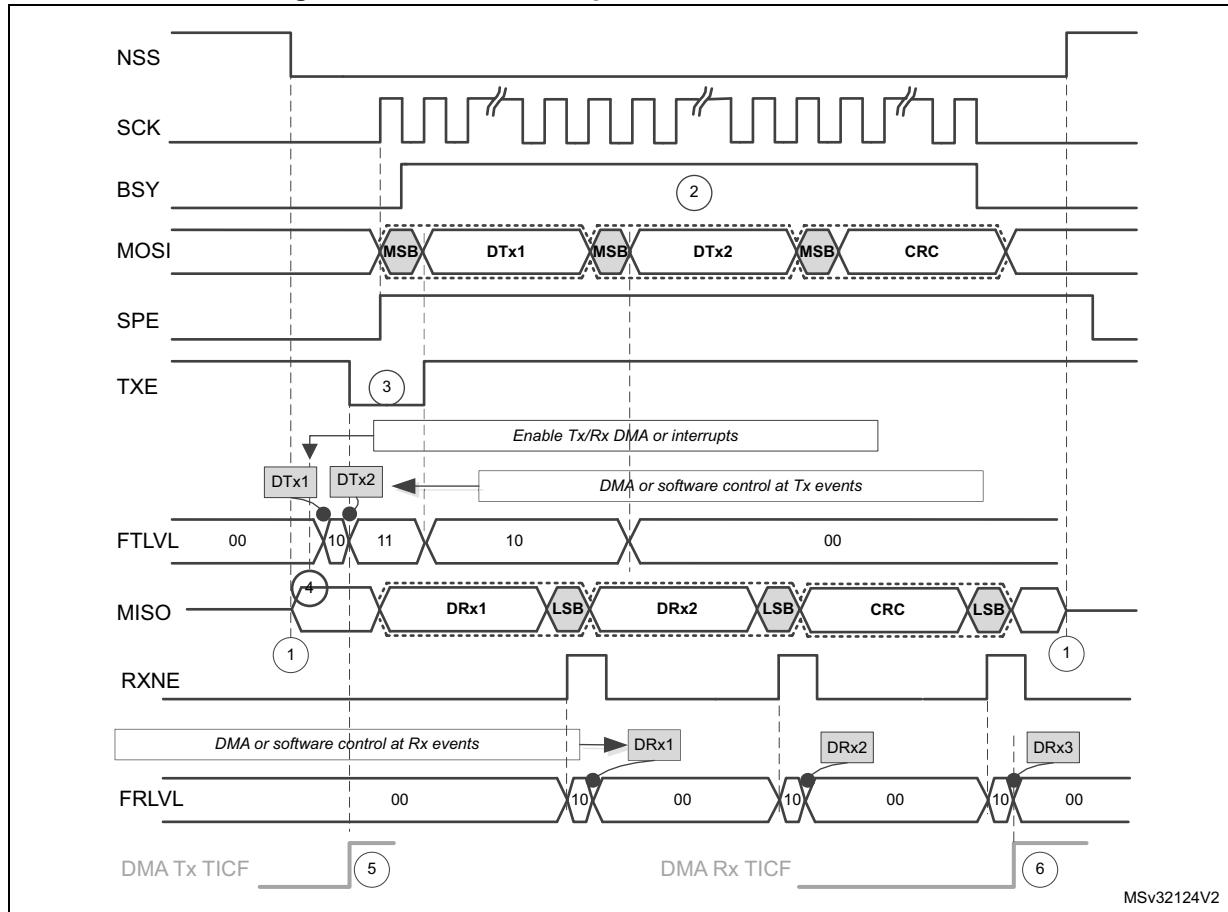
- Data size > 8 bit

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 762](#) for details about common assumptions and notes.

Figure 280. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

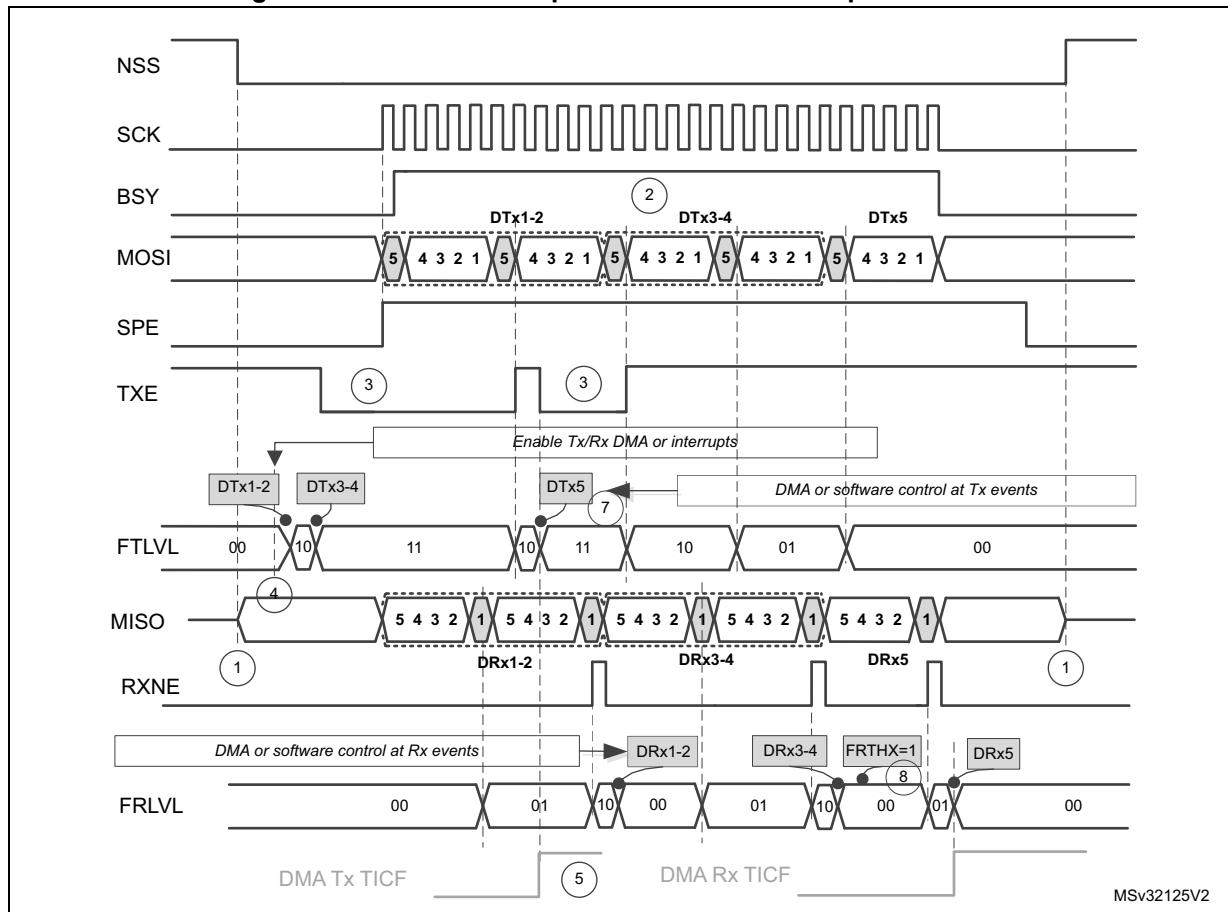
- Data size = 16 bit
- CRC enabled

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 762](#) for details about common assumptions and notes.

Figure 281. 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 762](#) for details about common assumptions and notes.

### 25.5.10 SPI status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

#### Tx buffer empty flag (TXE)

The TXE flag is set when transmission TXFIFO has enough space to store data to send. TXE flag is linked to the TXFIFO level. The flag goes high and stays high until the TXFIFO level is lower or equal to 1/2 of the FIFO depth. An interrupt can be generated if the TXEIE bit in the SPIx\_CR2 register is set. The bit is cleared automatically when the TXFIFO level becomes greater than 1/2.

#### Rx buffer not empty (RXNE)

The RXNE flag is set depending on the FRXTH bit value in the SPIx\_CR2 register:

- If FRXTH is set, RXNE goes high and stays high until the RXFIFO level is greater or equal to 1/4 (8-bit).
- If FRXTH is cleared, RXNE goes high and stays high until the RXFIFO level is greater than or equal to 1/2 (16-bit).

An interrupt can be generated if the RXNEIE bit in the SPIx\_CR2 register is set.

The RXNE is cleared by hardware automatically when the above conditions are no longer true.

#### Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect).

When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).

The BSY flag can be used in certain modes to detect the end of a transfer so that the software can disable the SPI or its peripheral clock before entering a low-power mode which does not provide a clock for the peripheral. This avoids corrupting the last transfer.

The BSY flag is also useful for preventing write collisions in a multimaster system.

The BSY flag is cleared under any one of the following conditions:

- When the SPI is correctly disabled
- When a fault is detected in Master mode (MODF bit set to 1)
- In Master mode, when it finishes a data transmission and no new data is ready to be sent
- In Slave mode, when the BSY flag is set to '0' for at least one SPI clock cycle between each data transfer.

Note:

*When the next transmission can be handled immediately by the master (e.g. if the master is in Receive-only mode or its Transmit FIFO is not empty), communication is continuous and the BSY flag remains set to '1' between transfers on the master side. Although this is not the case with a slave, it is recommended to use always the TXE and RXNE flags (instead of the BSY flags) to handle data transmission or reception operations.*

### 25.5.11 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERRIE bit.

#### Overrun flag (OVR)

An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RXFIFO) or when space for data storage is limited e.g. the RXFIFO is not available when CRC is enabled in receive only mode so in this case the reception buffer is limited into a single data frame buffer (see [Section 25.5.14: CRC calculation](#)).

When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost. Clearing the OVR bit is done by a read access to the SPI\_DR register followed by a read access to the SPI\_SR register.

#### Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit. Master mode fault affects the SPI interface in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPIx\_SR register while the MODF bit is set.
2. Then write to the SPIx\_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence. As a security, hardware does not allow the SPE and MSTR bits to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multimaster conflict.

#### CRC error (CRCERR)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPIx\_CR1 register is set. The CRCERR flag in the SPIx\_SR register is set if the value received in the shift register does not match the receiver SPIx\_RXCRCR value. The flag is cleared by the software.

#### TI mode frame format error (FRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPIx\_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of two data bytes.

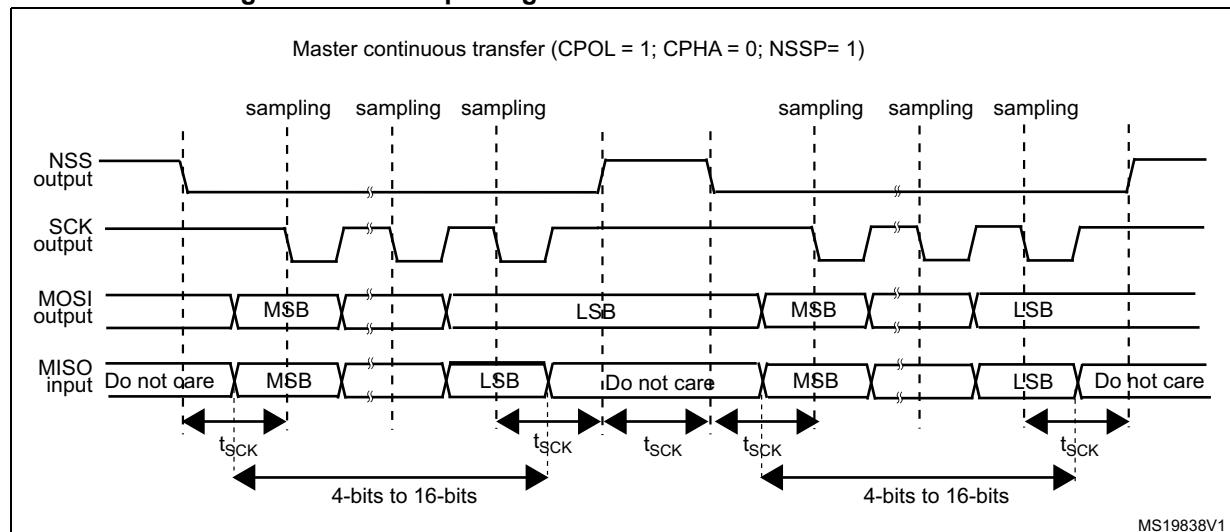
The FRE flag is cleared when SPIx\_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

### 25.5.12 NSS pulse mode

This mode is activated by the NSSP bit in the SPIx\_CR2 register and it takes effect only if the SPI interface is configured as Motorola SPI master (FRF=0) with capture on the first edge (SPIx\_CR1 CPHA = 0, CPOL setting is ignored). When activated, an NSS pulse is generated between two consecutive data frame transfers when NSS stays at high level for the duration of one clock period at least. This mode allows the slave to latch data. NSSP pulse mode is designed for applications with a single master-slave pair.

*Figure 282* illustrates NSS pin management when NSSP pulse mode is enabled.

**Figure 282. 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.

### 25.5.13 TI mode

#### TI protocol in master mode

The SPI interface is compatible with the TI protocol. The FRF bit of the SPIx\_CR2 register can be used to configure the SPI to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPIx\_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPIx\_CR1 and SPIx\_CR2 registers (SSM, SSI, SSOE) impossible in this case.

In slave mode, the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ when the current transaction finishes (see *Figure 283*). 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:

$$\frac{t_{\text{baud\_rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud\_rate}}}{2} + 6 \times t_{\text{pclk}}$$

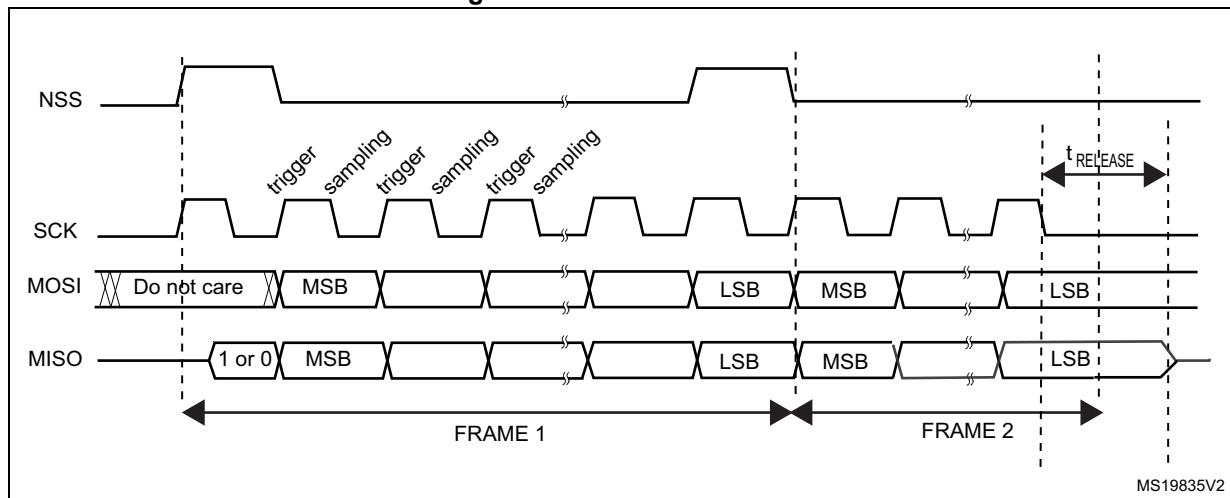
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

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 283: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

**Figure 283. TI mode transfer**



### 25.5.14 CRC calculation

Two separate CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers CRC8 or CRC16 calculation independently of the frame data length, which can be fixed to 8-bit or 16-bit. For all the other data frame lengths, no CRC is available.

#### CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx\_CR1 register before the SPI is enabled (SPE = 1). The CRC value is calculated using an odd programmable polynomial on each bit. The calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx\_CR1 register. The calculated CRC value is checked automatically at the end of the data block as well as for transfer managed by CPU or by the DMA. When a mismatch is detected between the CRC calculated internally on the received data and the CRC sent by the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC calculation depends on the SPI configuration and the chosen transfer management.

**Note:** *The polynomial value should only be odd. No even values are supported.*

### CRC transfer managed by CPU

Communication starts and continues normally until the last data frame has to be sent or received in the SPIx\_DR register. Then CRCNEXT bit has to be set in the SPIx\_CR1 register to indicate that the CRC frame transaction follows after the transaction of the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transaction. CRC calculation is frozen during CRC transaction.

The received CRC is stored in the RXFIFO like a data byte or word. That is why in CRC mode only, the reception buffer has to be considered as a single 16-bit buffer used to receive only one data frame at a time.

A CRC-format transaction usually takes one more data frame to communicate at the end of data sequence. However, when setting an 8-bit data frame checked by 16-bit CRC, two more frames are necessary to send the complete CRC.

When the last CRC data is received, an automatic check is performed comparing the received value and the value in the SPIx\_RXCRC register. Software has to check the CRCERR flag in the SPIx\_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing '0' to it.

After the CRC reception, the CRC value is stored in the RXFIFO and must be read in the SPIx\_DR register in order to clear the RXNE flag.

### CRC transfer managed by DMA

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication is automatic (with the exception of reading CRC data in receive only mode). The CRCNEXT bit does not have to be handled by the software. The counter for the SPI transmission DMA channel has to be set to the number of data frames to transmit excluding the CRC frame. On the receiver side, the received CRC value is handled automatically by DMA at the end of the transaction but user must take care to flush out received CRC information from RXFIFO as it is always loaded into it. In full-duplex mode, the counter of the reception DMA channel can be set to the number of data frames to receive including the CRC, which means, for example, in the specific case of an 8-bit data frame checked by 16-bit CRC:

$$\text{DMA\_RX} = \text{Numb\_of\_data} + 2$$

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 interface is configured as a slave, the NSS internal signal needs to be kept low during transaction of the CRC phase once the CRCNEXT signal is released. That is why the CRC calculation cannot be used at NSS Pulse mode when NSS hardware mode should be applied at slave normally.*

*At TI mode, despite the fact that clock phase and clock polarity setting is fixed and independent on SPIx\_CR1 register, the corresponding setting CPOL=0 CPHA=1 has to be kept at the SPIx\_CR1 register anyway if CRC is applied. In addition, the CRC calculation has to be reset between sessions by SPI disable sequence with re-enable the CRCEN bit described above at both master and slave side, else CRC calculation can be corrupted at this specific mode.*

## 25.6 SPI interrupts

During SPI communication an interrupt can be generated by the following events:

- Transmit TXFIFO ready to be loaded
- Data received in Receive RXFIFO
- Master mode fault
- Overrun error
- TI frame format error
- CRC protocol error

Interrupts can be enabled and disabled separately.

**Table 115. SPI interrupt requests**

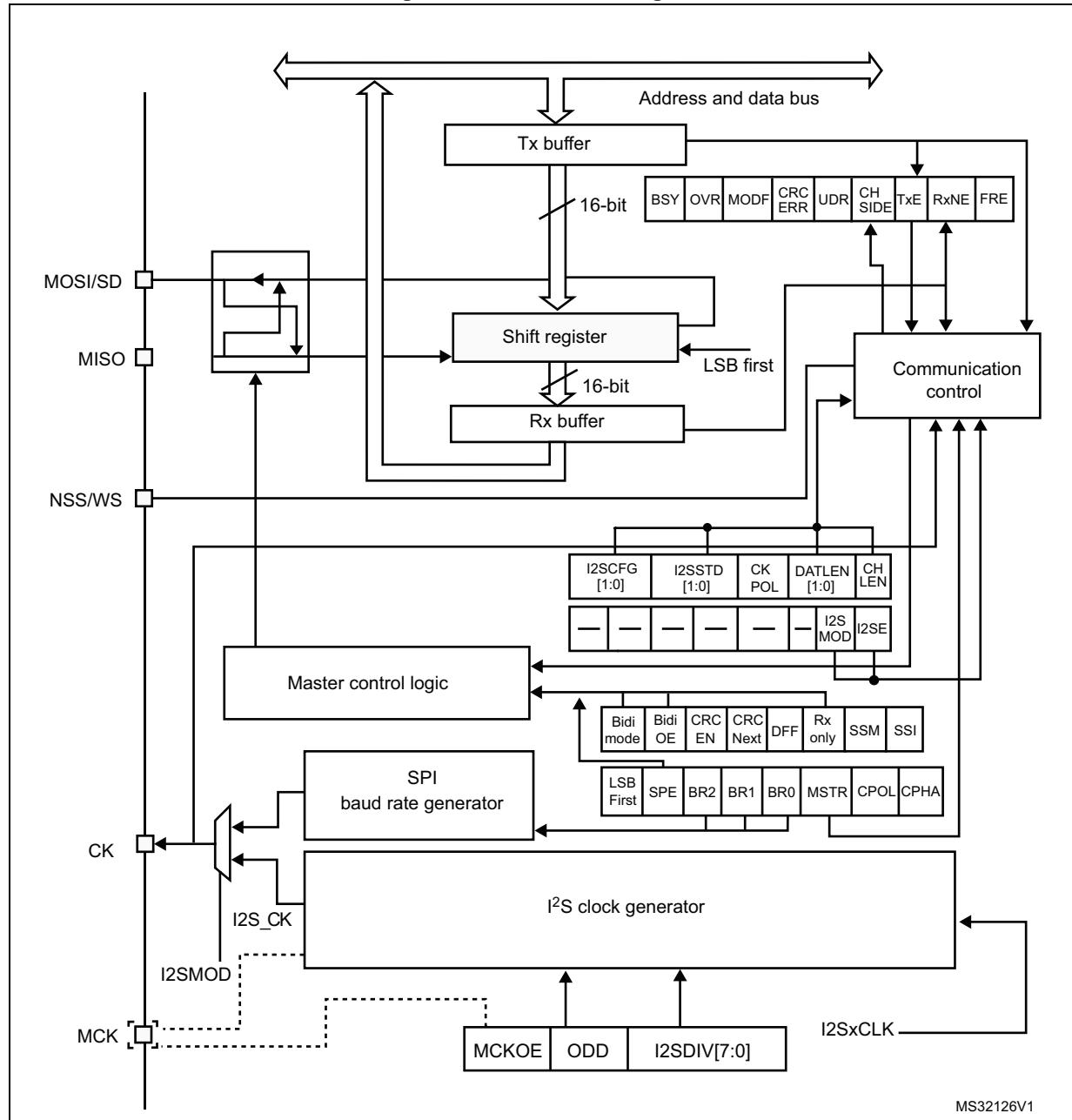
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

## 25.7 I2S functional description

### 25.7.1 I2S general description

The block diagram of the I2S is shown in [Figure 284](#).

**Figure 284. I2S block diagram**



1. MCK is mapped on the MISO pin.

The SPI can function as an audio I2S interface when the I2S capability is enabled (by setting the I2SMOD bit in the SPIx\_I2SCFGR register). This interface mainly uses the same pins, flags and interrupts as the SPI.

The I2S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I2S is configured in master mode (and when the MCKOE bit in the SPIx\_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to  $256 \times f_S$  for all I2S modes, and to  $128 \times f_S$  for all PCM modes, where  $f_S$  is the audio sampling frequency.

The I2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I<sup>2</sup>S mode. One is linked to the clock generator configuration SPIx\_I2SPR and the other one is a generic I2S configuration register SPIx\_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPIx\_CR1 register and all CRC registers are not used in the 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 I2S uses the same SPI register for data transfer (SPIx\_DR) in 16-bit wide mode.

## 25.7.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for transmission or reception. So, it is up to the software to write into the data register the appropriate value corresponding to each channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPIx\_SR register. Channel left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in a 16-bit frame
- 16-bit data packed in a 32-bit frame
- 24-bit data packed in a 32-bit frame
- 32-bit data packed in a 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPIx\_DR register or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 non-significant bits are extended to 32 bits with 0-bits (by hardware).

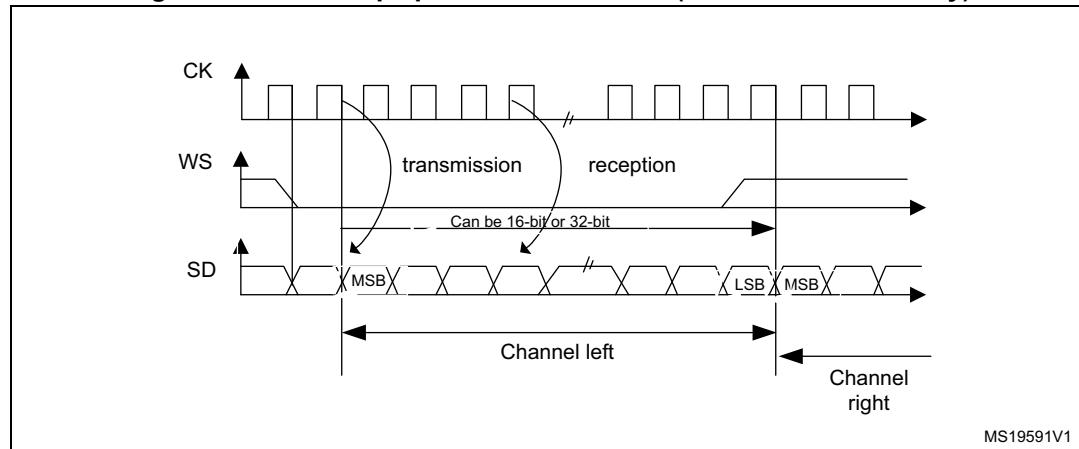
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The 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

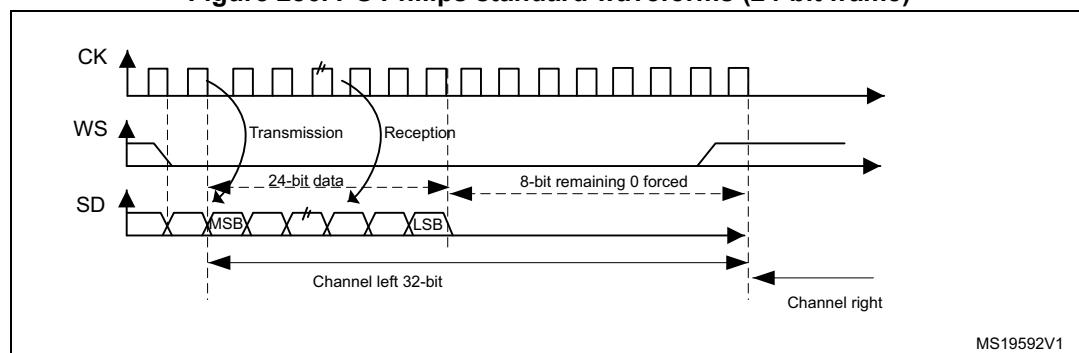
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

**Figure 285. I<sup>2</sup>S Philips protocol waveforms (16/32-bit full accuracy)**



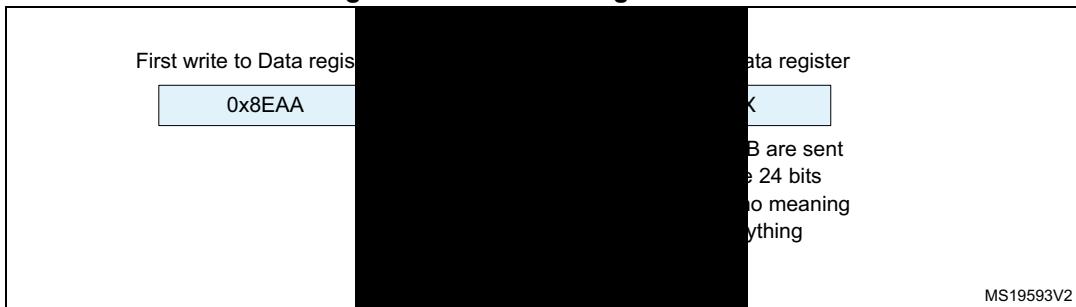
Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

**Figure 286. I<sup>2</sup>S Philips standard waveforms (24-bit frame)**

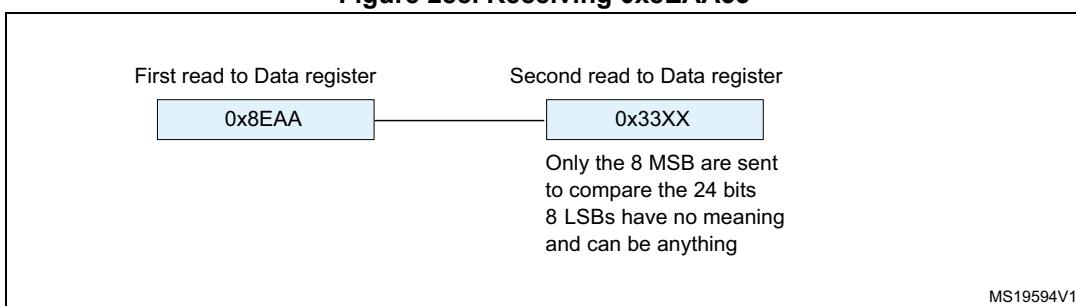
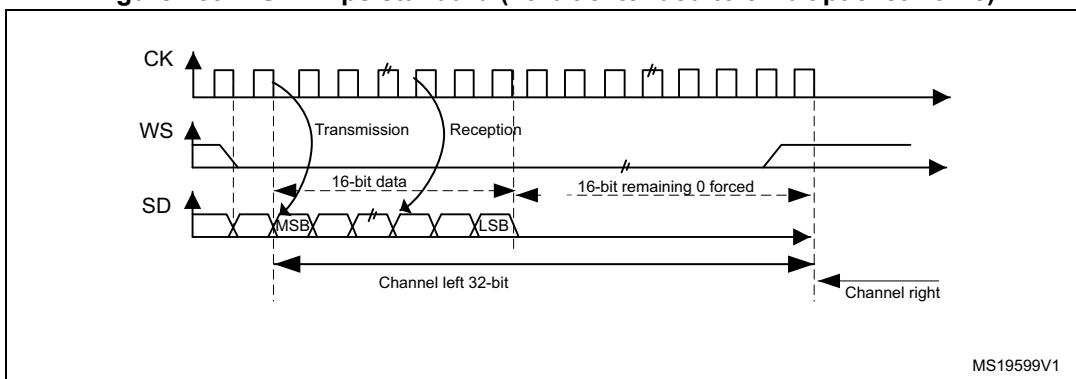


This mode needs two write or read operations to/from the SPIx\_DR register.

- In transmission mode:  
If 0x8EAA33 has to be sent (24-bit):

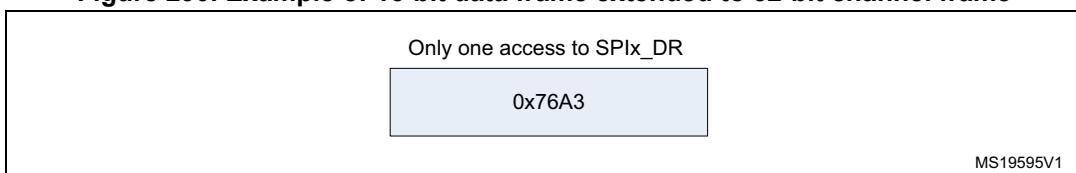
**Figure 287. Transmitting 0x8EAA33**

- In reception mode:  
If data 0x8EAA33 is received:

**Figure 288. Receiving 0x8EAA33****Figure 289. I<sup>2</sup>S Philips standard (16-bit extended to 32-bit packet frame)**

When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, only one access to the SPIx\_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in [Figure 290](#) is required.

**Figure 290. 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.

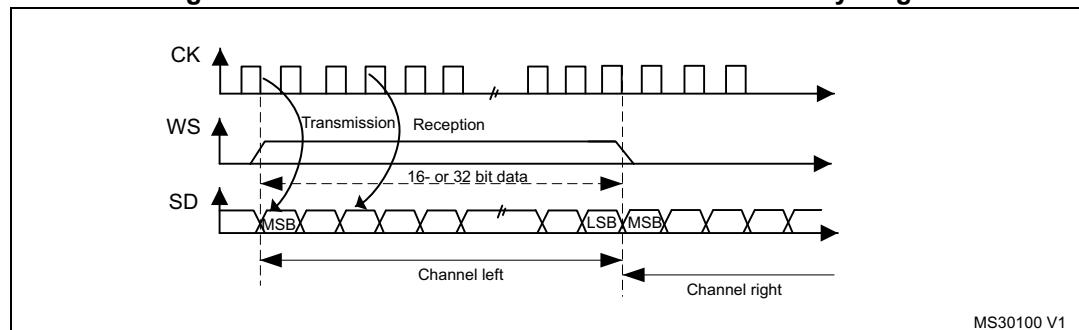
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

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

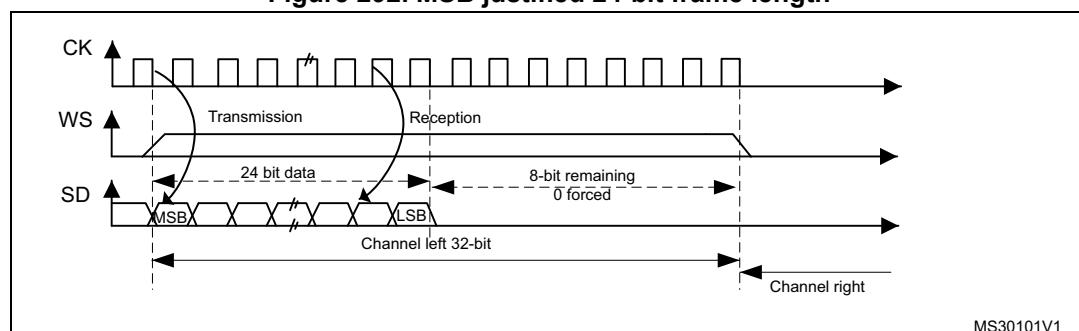
For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

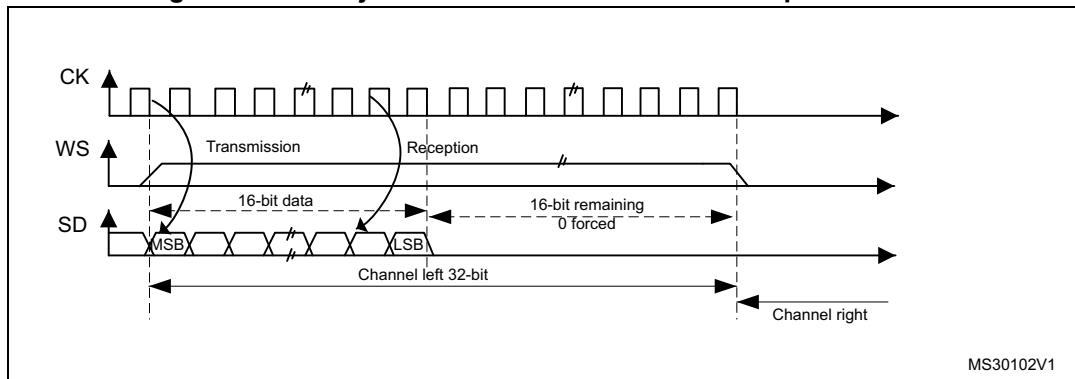
**Figure 291. MSB Justified 16-bit or 32-bit full-accuracy length**



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

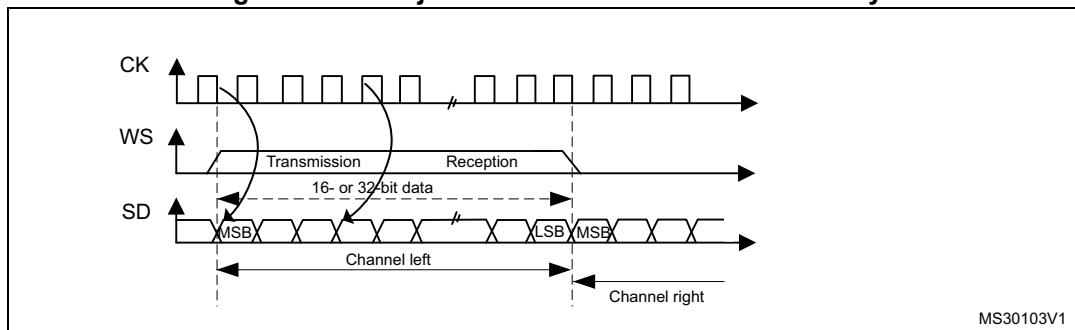
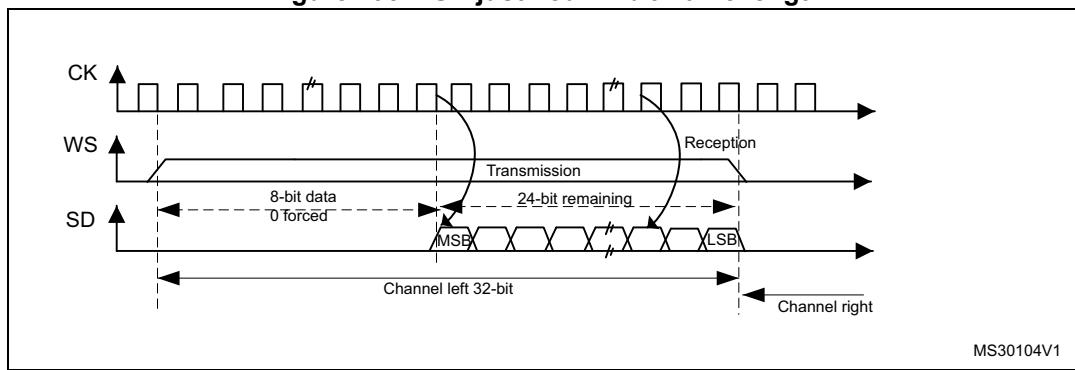
**Figure 292. MSB justified 24-bit frame length**



**Figure 293. MSB justified 16-bit extended to 32-bit packet frame****LSB justified standard**

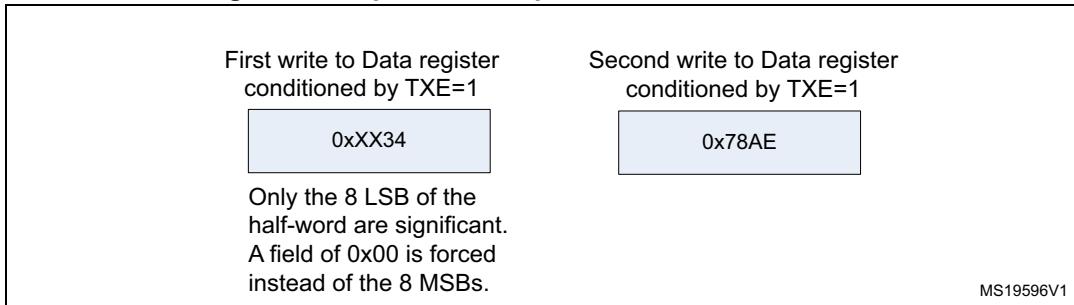
This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

The sampling of the input and output signals is the same as for the I<sup>2</sup>S Philips standard.

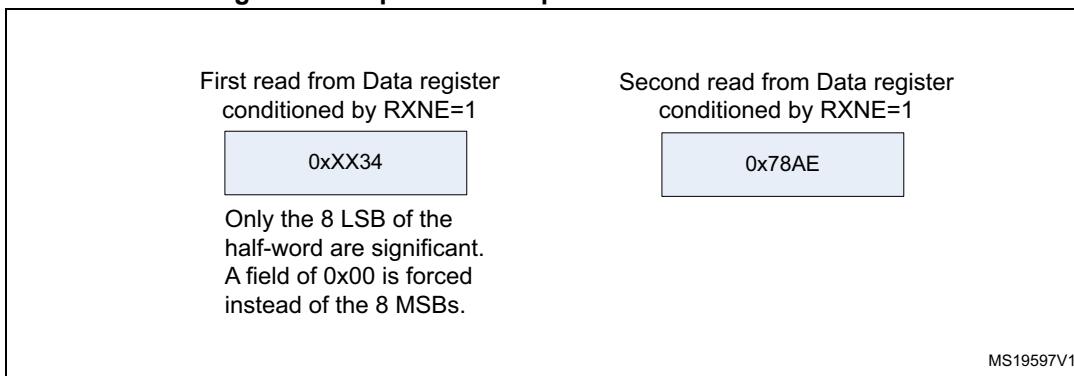
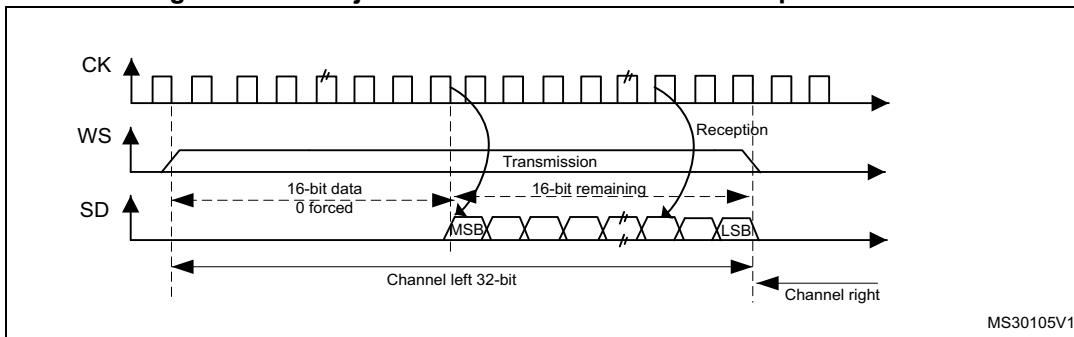
**Figure 294. LSB justified 16-bit or 32-bit full-accuracy****Figure 295. LSB justified 24-bit frame length**

- In transmission mode:

If data 0x3478AE have to be transmitted, two write operations to the SPIx\_DR register are required by software or by DMA. The operations are shown below.

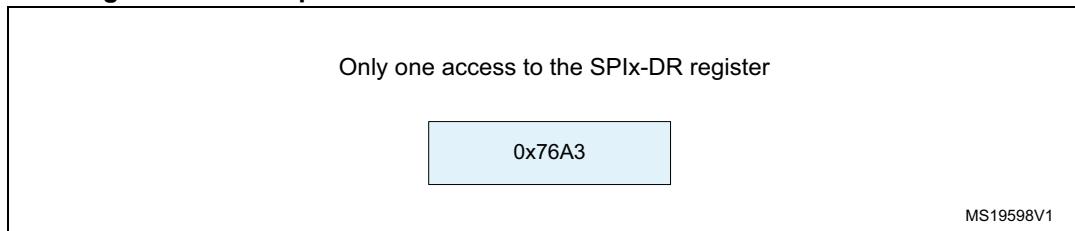
**Figure 296. Operations required to transmit 0x3478AE**

- In reception mode:  
If data 0x3478AE are received, two successive read operations from the SPIx\_DR register are required on each RXNE event.

**Figure 297. Operations required to receive 0x3478AE****Figure 298. LSB justified 16-bit extended to 32-bit packet frame**

When 16-bit data frame extended to 32-bit channel frame is selected during the I2S configuration phase, Only one access to the SPIx\_DR register is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in [Figure 299](#) is required.

**Figure 299. 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).

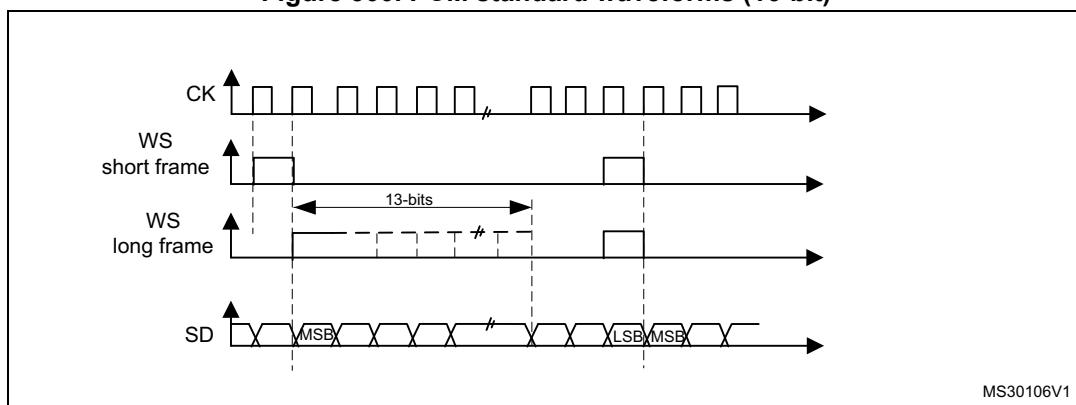
In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

### 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.

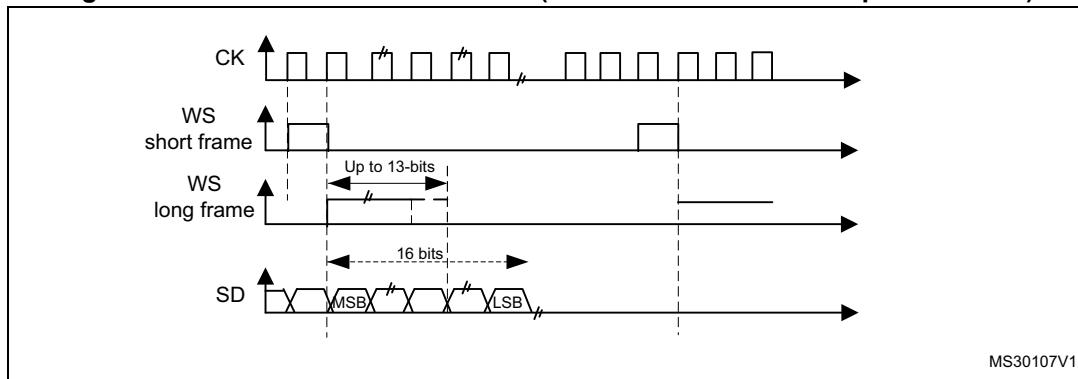
In PCM mode, the output signals (WS, SD) are sampled on the rising edge of CK signal. The input signals (WS, SD) are captured on the falling edge of CK.

Note that CK and WS are configured as output in MASTER mode.

**Figure 300. 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.

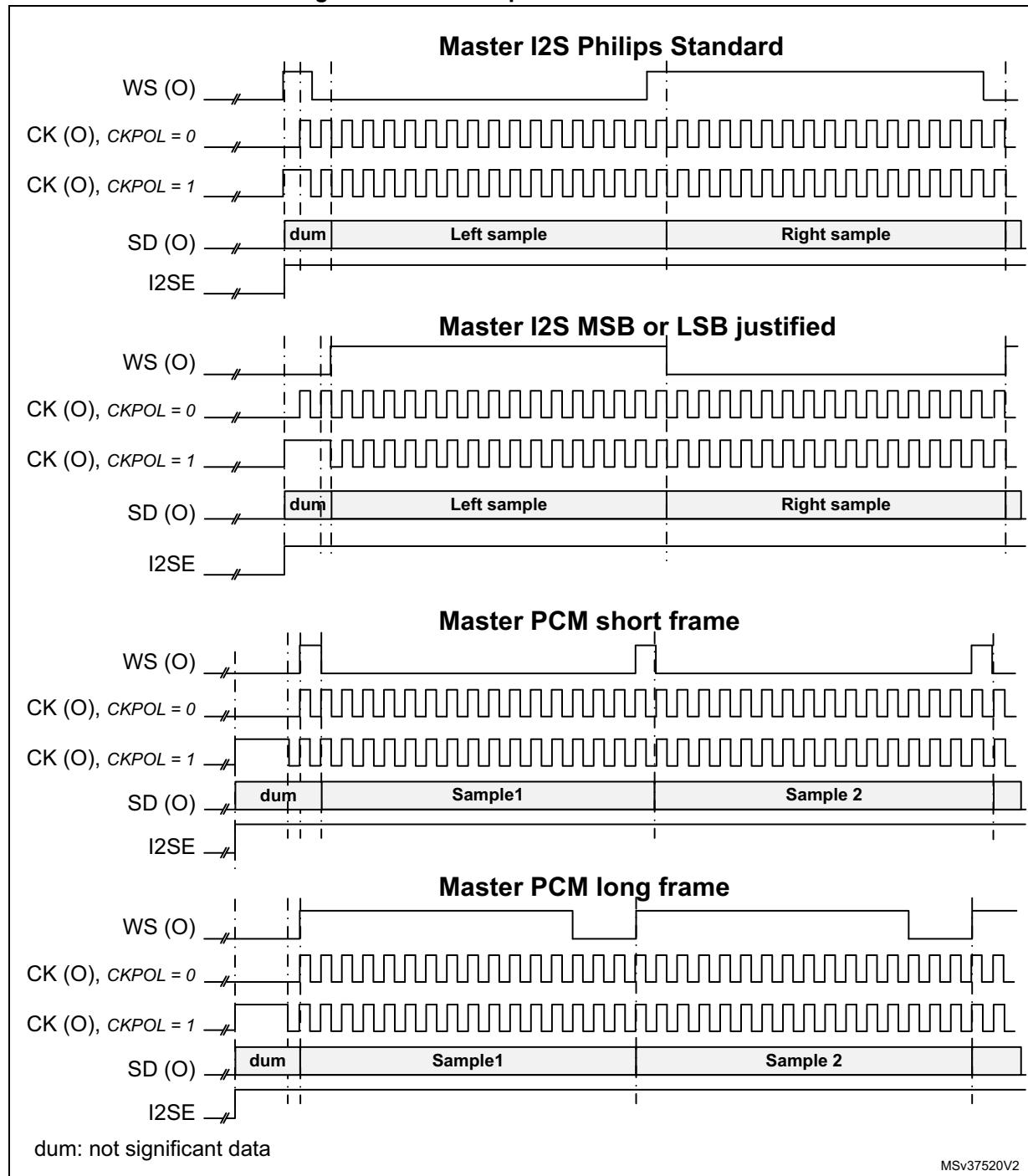
**Figure 301. PCM standard waveforms (16-bit extended to 32-bit packet frame)**

**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.

### 25.7.3 Start-up description

The [Figure 302](#) shows how the serial interface is handled in MASTER mode, when the SPI/I2S is enabled (via I2SE bit). It shows as well the effect of CKPOL on the generated signals.

Figure 302. Start sequence in master mode



In slave mode, the way the frame synchronization is detected, depends on the value of ASTRTEN bit.

If ASTRTEN = 0, when the audio interface is enabled (I2SE = 1), then the hardware waits for the appropriate transition on the incoming WS signal, using the CK signal.

The appropriate transition is a falling edge on WS signal when I<sup>2</sup>S Philips Standard is used, or a rising edge for other standards. The falling edge is detected by sampling first WS to 1 and then to 0, and vice-versa for the rising edge detection.

If ASTRTEN = 1, the user has to enable the audio interface before the WS becomes active. This means that the I2SE bit must be set to 1 when WS = 1 for I<sup>2</sup>S Philips standard, or when WS = 0 for other standards.

#### 25.7.4 Clock generator

The I<sup>2</sup>S bit rate determines the data flow on the I<sup>2</sup>S data line and the I<sup>2</sup>S clock signal frequency.

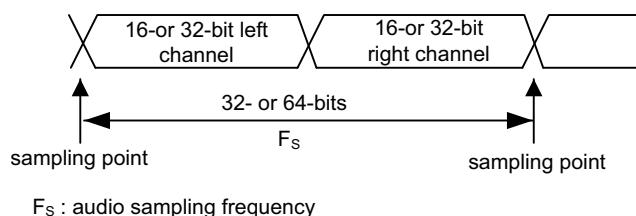
I<sup>2</sup>S bit rate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I<sup>2</sup>S bit rate is calculated as follows:

$$\text{I}^2\text{S bit rate} = 16 \times 2 \times f_S$$

It is: I<sup>2</sup>S bit rate = 32 × 2 × f<sub>S</sub> if the packet length is 32-bit wide.

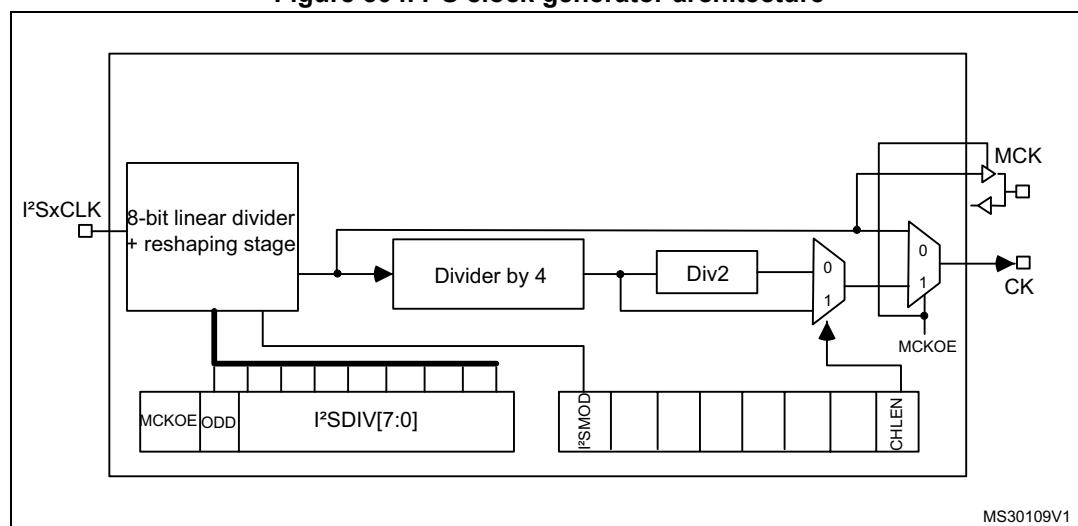
**Figure 303. Audio sampling frequency definition**



MS30108V1

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 304. I<sup>2</sup>S clock generator architecture**



MS30109V1

1. Where x can be 2 or 3.

*Figure 304* presents the communication clock architecture. The I2SxCLK clock is provided by the reset and clock controller (RCC) of the product. The I2SxCLK clock can be asynchronous with respect to the SPI/I2S APB clock.

---

**Warning:** In addition, it is mandatory to keep the I2SxCLK frequency higher or equal to the APB clock used by the SPI/I2S block. If this condition is not respected the SPI/I2S does not work properly.

---

The audio sampling frequency may be 192 kHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range).

In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

#### For I<sup>2</sup>S modes:

When the master clock is generated (MCKOE in the SPIx\_I2SPR register is set):

$$F_S = \frac{F_{I2SxCLK}}{256 \times ((2 \times I2SDIV) + ODD)}$$

When the master clock is disabled (MCKOE bit cleared):

$$F_S = \frac{F_{I2SxCLK}}{32 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}$$

CHLEN = 0 when the channel frame is 16-bit wide and,

CHLEN = 1 when the channel frame is 32-bit wide.

#### For PCM modes:

When the master clock is generated (MCKOE in the SPIx\_I2SPR register is set):

$$F_S = \frac{F_{I2SxCLK}}{128 \times ((2 \times I2SDIV) + ODD)}$$

When the master clock is disabled (MCKOE bit cleared):

$$F_S = \frac{F_{I2SxCLK}}{16 \times (CHLEN + 1) \times ((2 \times I2SDIV) + ODD)}$$

CHLEN = 0 when the channel frame is 16-bit wide and,

CHLEN = 1 when the channel frame is 32-bit wide.

Where  $F_S$  is the audio sampling frequency, and  $F_{I2SxCLK}$  is the frequency of the kernel clock provided to the SPI/I2S block.

**Note:** I<sub>2</sub>S DIV must be strictly higher than 1.

The following table provides example precision values for different clock configurations.

**Note:** Other configurations are possible that allow optimum clock precision.

**Table 116. Audio-frequency precision using 48 MHz clock derived from HSE<sup>(1)</sup>**

SYSCLK (MHz)	Data length	I <sub>2</sub> S DIV	I <sub>2</sub> S ODD	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.

## 25.7.5 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 25.7.4: 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 TXIE 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 25.7.2: 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 25.7.5: I<sup>2</sup>S 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 25.7.2: 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.

### 25.7.6 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 I2S interface. The clock and WS signals are input from the external master connected to the I2S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPIx\_I2SCFGR register to select 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.

### Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the NSS\_WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I2S data register has to be loaded before the master initiates the communication.

For the I2S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I2S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

*Note:* *The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.*

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx\_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I<sup>2</sup>S standard mode selected, refer to [Section 25.7.2: Supported audio protocols](#).

To secure a continuous audio data transmission, it is mandatory to write the SPIx\_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPIx\_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPIx\_CR2 register, an interrupt is generated when the UDR flag in the SPIx\_SR register goes high. In this case, it is mandatory to switch off the I2S and to restart a data transfer starting from the left channel.

To switch off the I2S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

## Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in [Section 25.7.6: I<sup>2</sup>S 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 25.7.2: 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.*

## 25.7.7 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 I2S is disabled (I2SE bit is reset).

### RX buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPIx\_DR register is read.

### Channel Side flag (CHSIDE)

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I2S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPIx\_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I2S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

When the OVR or UDR flag in the SPIx\_SR is set and the ERRIE bit in SPIx\_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx\_SR status register (once the interrupt source has been cleared).

## 25.7.8 I2S error flags

There are three error flags for the I2S cell.

### Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx\_DR. It is available when the I2SMOD bit in the SPIx\_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx\_CR2 register is set.

The UDR bit is cleared by a read operation on the SPIx\_SR register.

### Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from the SPIx\_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx\_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx\_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx\_DR register followed by a read access to the SPIx\_SR register.

### Frame error flag (FRE)

This flag can be set by hardware only if the I2S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the

synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the 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 CK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

### 25.7.9 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.

## 25.8 I<sup>2</sup>S interrupts

*Table 117* provides the list of I<sup>2</sup>S interrupts.

**Table 117. 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	

## 25.9 SPI and I2S registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit). SPI\_DR in addition can be accessed by 8-bit access.

### 25.9.1 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	BIDIOE	CRC EN	CRCN EXT	CRCL	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR[2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **BIDI MODE**: 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 BIDI MODE 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 **CRCEN**: Hardware CRC calculation enable

0: CRC calculation disabled  
1: CRC calculation enabled

*Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.  
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 758](#).*

*This bit is not used in I<sup>2</sup>S mode.*

Bits 5:3 **BR[2:0]:** Baud rate control

- 000: f<sub>PCLK</sub>/2
- 001: f<sub>PCLK</sub>/4
- 010: f<sub>PCLK</sub>/8
- 011: f<sub>PCLK</sub>/16
- 100: f<sub>PCLK</sub>/32
- 101: f<sub>PCLK</sub>/64
- 110: f<sub>PCLK</sub>/128
- 111: f<sub>PCLK</sub>/256

*Note: These bits should not be changed when communication is ongoing.*

*These bits are 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.*

Bit 1 **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 except the case when CRC is applied at 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 except the case when CRC is applied at TI mode.*

### 25.9.2 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	FRXTH	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 758](#) 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 758](#) 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.*

Bits 11:8 **DS[3:0]**: Data size

These bits configure the data length for SPI transfers.

- 0000: Not used
- 0001: Not used
- 0010: Not used
- 0011: 4-bit
- 0100: 5-bit
- 0101: 6-bit
- 0110: 7-bit
- 0111: 8-bit
- 1000: 9-bit
- 1001: 10-bit
- 1010: 11-bit
- 1011: 12-bit
- 1100: 13-bit
- 1101: 14-bit
- 1110: 15-bit
- 1111: 16-bit

If software attempts to write one of the “Not used” values, they are forced to the value “0111” (8-bit)

*Note: These bits are 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

### 25.9.3 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[1:0]		FRE	BSY	OVR	MODF	CRCE RR	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: This bit is 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.*

Bit 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 25.5.11: SPI error flags](#) and [Section 25.7.8: I2S error flags](#).

This flag is set by hardware and reset when SPIx\_SR is read by software.

0: No frame format error

1: A frame format error occurred

Bit 7 **BSY:** Busy flag

- 0: SPI (or I2S) not busy
  - 1: SPI (or I2S) is busy in communication or Tx buffer is not empty
- This flag is set and cleared by hardware.

*Note: The BSY flag must be used with caution: refer to [Section 25.5.10: SPI status flags and Procedure for disabling the SPI](#) on page 758.*

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 790](#) 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 768](#) 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

*Note: This flag is set by hardware and cleared by software writing 0.*

*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 790](#) 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

### 25.9.4 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 TxFIFO (See [Section 25.5.9: Data transmission and reception procedures](#)).

*Note: Data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read. The Rx threshold setting must always correspond with the read access currently used.*

### 25.9.5 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 (0x0007) is the reset value of this register. Another polynomial can be configured as required.

*Note: The polynomial value should be odd only. No even value is supported.*

### 25.9.6 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 CRC frame format is set to be 8-bit length (CRCL bit in the SPIx\_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx\_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note: A read to this register when the BSY Flag is set could return an incorrect value.*

*These bits are not used in I<sup>2</sup>S mode.*

**25.9.7 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 CRC frame format is set to be 8-bit length (CRCL bit in the SPIx\_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit CRC frame format is selected (CRCL bit in the SPIx\_CR1 register is set). CRC calculation is done based on any CRC16 standard.

*Note: A read to this register when the BSY flag is set could return an incorrect value.*

*These bits are not used in I<sup>2</sup>S mode.*

**25.9.8 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.	ASTR TEN	I2SMOD	I2SE	I2SCFG[1:0]		PCMSYNC	Res.	I2SSSTD[1:0]		CKPOL	DATLEN[1:0]		CHLEN
			rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ASTRTEM**: Asynchronous start enable.

0: The Asynchronous start is disabled.

When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is received and an appropriate transition is detected on the WS signal.

1: The Asynchronous start is enabled.

When the I2S is enabled in slave mode, the hardware starts the transfer when the I2S clock is received and the appropriate level is detected on the WS signal.

*Note: The appropriate **transition** is a falling edge on WS signal when I<sup>2</sup>S Philips Standard is used, or a rising edge for other standards.*

*The appropriate **level** is a low level on WS signal when I<sup>2</sup>S Philips Standard is used, or a high level for other standards.*

*Please refer to [Section 25.7.3: Start-up description](#) for additional information.*

Bit 11 **I2SMOD**: I2S mode selection

0: SPI mode is selected

1: I2S mode is selected

*Note: This bit should be configured when the SPI is disabled.*

Bit 10 **I2SE**: I2S enable

0: I2S peripheral is disabled

1: I2S peripheral is enabled

*Note: This bit is not used in SPI mode.*

Bits 9:8 **I2SCFG[1:0]**: I2S configuration mode

00: Slave - transmit

01: Slave - receive

10: Master - transmit

11: Master - receive

*Note: These bits should be configured when the I2S is disabled.*

*They are not used in SPI mode.*

Bit 7 **PCMSYNC**: PCM frame synchronization

0: Short frame synchronization

1: Long frame synchronization

*Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used).*

*It is not used in SPI mode.*

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I2S standard selection

00: I<sup>2</sup>S Philips standard

01: MSB justified standard (left justified)

10: LSB justified standard (right justified)

11: PCM standard

For more details on I<sup>2</sup>S standards, refer to [Section 25.7.2 on page 774](#)

*Note: For correct operation, these bits should be configured when the I2S is disabled.*

*They are not used in SPI mode.*

Bit 3 **CKPOL**: Inactive state clock polarity

- 0: I2S clock inactive state is low level
- 1: I2S clock inactive state is high level

*Note: For correct operation, this bit should be configured when the I2S is disabled.*

*It is not used in SPI mode.*

*The bit CKPOL does not affect the CK edge sensitivity used to receive or transmit the SD and WS signals.*

Bits 2:1 **DATLEN[1:0]**: Data length to be transferred

- 00: 16-bit data length
- 01: 24-bit data length
- 10: 32-bit data length
- 11: Not allowed

*Note: For correct operation, these bits should be configured when the I2S is disabled.*

*They are not used in SPI mode.*

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)

- 0: 16-bit wide
- 1: 32-bit wide

The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in.

*Note: For correct operation, this bit should be configured when the I2S is disabled.*

*It is not used in SPI mode.*

### 25.9.9 SPIx\_I2S prescaler register (SPIx\_I2SPR)

Address offset: 0x20

Reset value: 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[7:0]							

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

Bit 9 **MCKOE**: Master clock output enable

- 0: Master clock output is disabled
- 1: Master clock output is enabled

*Note: This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode.*

*It is not used in SPI mode.*

Bit 8 **ODD**: Odd factor for the prescaler

- 0: Real divider value is = I2SDIV \*2
  - 1: Real divider value is = (I2SDIV \* 2) + 1
- Refer to [Section 25.7.3 on page 781](#).

*Note: This bit should be configured when the I2S is disabled. It is used only when the I2S is in master mode.*

*It is not used in SPI mode.*

Bits 7:0 **I2SDIV[7:0]**: I2S linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to [Section 25.7.3 on page 781](#).

*Note: These bits should be configured when the I2S is disabled. They are used only when the I2S is in master mode.*

*They are not used in SPI mode.*

## 25.9.10 SPI/I2S register map

*Table 118* shows the SPI/I2S register map and reset values.

**Table 118. SPI/I2S register map and reset values**

Offset	Register name reset value	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPIx_CR1	BIDIMODE															
	Reset value	0	0	LDMA_RX	0	LDMA_TX	0	0	0	0	0	0	0	0	0	0	0
0x04	SPIx_CR2	Res.	Res.	Res.	Res.	DS[3:0]	0	0	0	0	0	0	0	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	SPIx_SR	Res.	Res.	Res.	Res.	FTLVL[1:0]	0	1	1	1	BSY	TXEIE	LSBFIRST	0	0	0	0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SPIx_DR	DR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPIx_CRCPR	CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SPIx_RXCRCR	RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPIx_TXCRCR	TXCRC[15:0]															
	Reset value	0	0	0	0	I2SMOD	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPIx_I2SCFGR	Res.	Res.	Res.	Res.	ASTRTEN	0	I2SE	I2SSTD	CKPOL	DATLEN[1:0]	0	0	0	0	0	0
	Reset value	Res.	Res.	Res.	Res.	MCKOE	0	I2SCFG[1:0]	PCM SYNC	0	0	0	0	0	0	0	0
0x20	SPIx_I2SPR	Res.	Res.	Res.	Res.	ODD	0	0	0	0	0	I2SDIV[7:0]	0	0	0	0	0
	Reset value	Res.	Res.	Res.	Res.	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 26 Debug support (DBG)

### 26.1 Overview

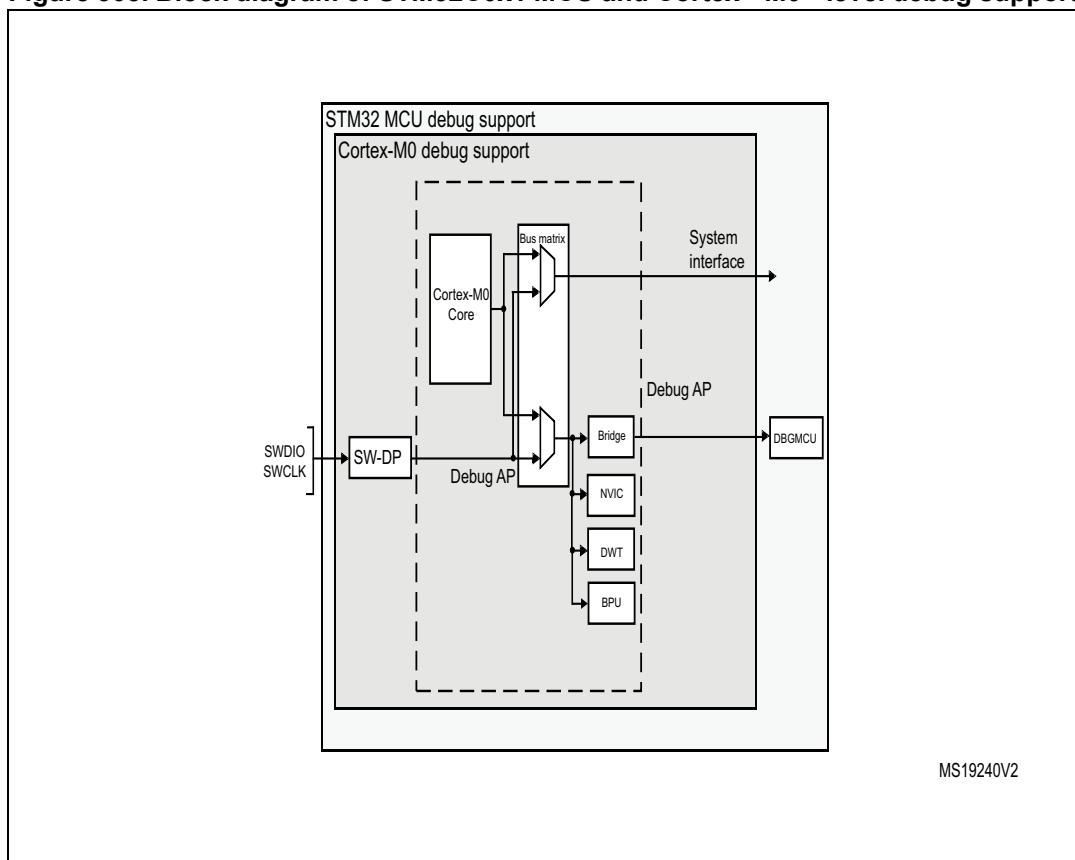
The STM32C0x1 devices are built around a Cortex<sup>®</sup>-M0+ core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32C0x1 MCUs.

One interface for debug is available:

- Serial wire

**Figure 305. Block diagram of STM32C0x1 MCU and Cortex<sup>®</sup>-M0+-level debug support**



1. The debug features embedded in the Cortex<sup>®</sup>-M0+ core are a subset of the Arm CoreSight Design Kit.

The Arm Cortex<sup>®</sup>-M0+ core provides integrated on-chip debug support. It is comprised of:

- SW-DP: Serial wire
- BPU: Break point unit
- DWT: Data watchpoint trigger

It also includes debug features dedicated to the STM32C0x1:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

**Note:** *For further information on debug functionality supported by the Arm Cortex®-M0+ core, refer to the Cortex®-M0+ Technical Reference Manual (see [Section 26.2: Reference Arm documentation](#)).*

## 26.2 Reference Arm documentation

- Cortex®-M0+ Technical Reference Manual (TRM), available from <http://infocenter.arm.com>
- Arm Debug Interface V5
- Arm CoreSight Design Kit revision r1p1 Technical Reference Manual

## 26.3 Pinout and debug port pins

The STM32C0x1 MCUs are available in various packages with different numbers of available pins.

### 26.3.1 SWD port pins

Two pins are used as outputs for the SW-DP as alternate functions of general purpose I/Os. These pins are available on all packages.

**Table 119. SW debug port pins**

SW-DP pin name	SW debug port		Pin assignment
	Type	Debug assignment	
SWDIO	I/O	Serial Wire Data Input/Output	PA13
SWCLK	I	Serial Wire Clock	PA14

### 26.3.2 SW-DP pin assignment

After reset (SYSRESETn or PORESETn), the pins used for the SW-DP are assigned as dedicated pins which are immediately usable by the debugger host.

However, the MCU offers the possibility to disable the SWD port and can then release the associated pins for general-purpose I/O (GPIO) usage. For more details on how to disable SW-DP port pins, refer to [Section 6.3.2: I/O pin alternate function multiplexer and mapping on page 148](#).

### 26.3.3 Internal pull-up & pull-down on SWD pins

Once the SW I/O is released by the user software, the GPIO controller takes control of these pins. The reset states of the GPIO control registers put the I/Os in the equivalent states:

- SWDIO: input pull-up
- SWCLK: input pull-down

*Having embedded pull-up and pull-down resistors removes the need to add external resistors.*

## 26.4 ID codes and locking mechanism

There are several ID codes inside the MCU. ST strongly recommends the tool manufacturers (for example Keil, IAR, Raisonance) to lock their debugger using the MCU device ID located at address 0x40015800.

Only the DEV\_ID[15:0] should be used for identification by the debugger/programmer tools (the revision ID must not be taken into account).

## 26.5 SWD port

### 26.5.1 SWD protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 kΩ recommended by Arm).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

### 26.5.2 SWD protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

**Table 120. 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 124 on page 809</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®-M0+ *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

**Table 121. 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 122. 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.

### 26.5.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default Arm one and is set to **0x0BB11477** (corresponding to Cortex®-M0+).

- Note:** Note that the SW-DP state machine is inactive until the target reads this ID code.
- The SW-DP state machine is in RESET STATE either after power-on reset, or after the line is high for more than 50 cycles
  - The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
  - After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target issues a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex®-M0+ TRM* and the *CoreSight Design Kit r1p0 TRM*.

#### 26.5.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.  
The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)  
This is particularly important when writing the CTRL/STAT for a power-on request. If the next transaction (requiring a power-on) occurs immediately, it fails.

#### 26.5.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 123. SW-DP registers

A[3:2]	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read		IDCODE	The manufacturer code is set to the default Arm code for Cortex®-M0+: <b>0x0BC11477</b> (identifies the SW-DP)
00	Write		ABORT	

**Table 123. SW-DP registers (continued)**

A[3:2]	R/W	CTRLSEL bit of SELECT register	Register	Notes
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-on – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-on acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read		READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write		SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write		READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

### 26.5.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register.

**Table 124. 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	DP CTRL/STAT register. Used to: – Request a system or debug power-on – Configure the transfer operation for AP accesses – Control the pushed compare and pushed verify operations. – Read some status flags (overrun, power-on acknowledges)

**Table 124. 32-bit debug port registers addressed through the shifted value A[3:2]**

Address	A[3:2] value	Description
0x8	10	DP SELECT register: Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

## 26.6 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the debug access port. It consists of four registers:

**Table 125. Core debug registers**

Register	Description
DHCSR	<i>The 32-bit Debug Halting Control and Status Register</i> This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	<i>The 17-bit Debug Core Register Selector Register:</i> This selects the processor register to transfer data to or from.
DCRDR	<i>The 32-bit Debug Core Register Data Register:</i> This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	<i>The 32-bit Debug Exception and Monitor Control Register:</i> This provides Vector Catching and Debug Monitor Control.

These registers are not reset by a system reset. They are only reset by a power-on reset. Refer to the Cortex®-M0+ TRM for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC\_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C\_DEBUGEN) of the Debug Halting Control and Status Register

## 26.7 BPU (break point unit)

The Cortex®-M0+ BPU implementation provides four breakpoint registers. The BPU is a subset of the Flash Patch and Breakpoint (FPB) block available in Armv7-M (Cortex-M3 & Cortex-M4).

### 26.7.1 BPU functionality

The processor breakpoints implement PC based breakpoint functionality.

Refer the Armv6-M Arm and the Arm CoreSight Components Technical Reference Manual for more information about the BPU CoreSight identification registers, and their addresses and access types.

## 26.8 DWT (data watchpoint)

The Cortex®-M0+ DWT implementation provides two watchpoint register sets.

### 26.8.1 DWT functionality

The processor watchpoints implement both data address and PC based watchpoint functionality, a PC sampling register, and support comparator address masking, as described in the *Armv6-M Arm*.

### 26.8.2 DWT Program counter sample register

A processor that implements the data watchpoint unit also implements the Armv6-M optional *DWT Program Counter Sample Register* (DWT\_PCSR). This register permits a debugger to periodically sample the PC without halting the processor. This provides coarse grained profiling. See the *Armv6-M Arm* for more information.

The Cortex®-M0+ DWT\_PCSR records both instructions that pass their condition codes and those that fail.

## 26.9 MCU debug component (DBG)

The MCU debug component helps the debugger provide support for:

- low-power modes
- clock control for timers, watchdog and I2C during a breakpoint

### 26.9.1 Debug support for low-power modes

The CPU requires active FCLK or HCLK clocks to allow any debug.

By default, Stop, Standby, and Shutdown low-power modes deactivate FCLK and HCLK, which prevents debug capability. In Sleep mode however, the device keeps FCLK and HCLK always active.

To keep FCLK or HCLK clocks active and so preserve debug capability in Stop, Standby, and Shutdown modes, the debugger host must set, before entering one of these low-power modes, the DBG\_STOP bit (for Stop) or DBG\_STANDBY bit (for Standby and Shutdown) of the DBG\_CR register.

### 26.9.2 Debug support for timers, watchdog, and I2C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I2C peripheral, the user can choose to block the SMBUS timeout during a breakpoint.

## 26.10 DBG registers

The devices integrate an ID code identifying the device and its die revision.

This code is accessible by the software debug port (two pins) or by the user software.

### 26.10.1 DBG device ID code register (DBG\_IDCODE)

Address offset: 0x00

Reset value: refer to [Table 126](#)

Only 32-bit access supported.

This read-only register allows identifying the device and its die revision. It is accessible through the software debug port (two pins) or the user software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[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
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV\_ID[15:0]**: Revision identifier

This field indicates the revision of the device. Refer to [Table 126](#).

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

Upon read, these reserved bits return 0b0110.

Bits 11:0 **DEV\_ID[11:0]**: Device identifier

This field indicates the device ID. Refer to [Table 126](#).

**Table 126. DEV\_ID and REV\_ID field values**

Device	DEV_ID	Revision code	Revision number	REV_ID
STM32C011xx	0x443	A	1.0	0x1000
		Z	1.1	0x1001
STM32C031xx	0x453	A	1.0	0x1000
		Z	1.1	0x1001

### 26.10.2 DBG configuration register (DBG\_CR)

Address offset: 0x0000 0004

Reset value: 0x0000 0000 (power-on reset)

Only 32-bit access supported.

This register configures the low-power modes of the MCU under debug.

It is asynchronously reset by the POR, but not affected by the system reset. It can be written by the debugger under system reset.

If the debugger host does not support this feature, it is still possible for the user software to write this register.

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.	DBG_STANDBY	DBG_STOP													
														rw	rw

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

Bit 2 **DBG\_STANDBY:** Debug Standby and Shutdown modes

Debug options in Standby or Shutdown mode.

0: Digital part powered. From software point of view, exiting Standby and Shutdown modes is identical as fetching reset vector (except for status bits indicating that the MCU exits Standby)

1: Digital part powered and FCLK and HCLK running, derived from the internal RC oscillator remaining active. The MCU generates a system reset so that exiting Standby and Shutdown has the same effect as starting from reset.

Bit 1 **DBG\_STOP:** Debug Stop mode

Debug options in Stop mode.

0: All clocks disabled, including FCLK and HCLK. Upon Stop mode exit, the CPU is clocked by the HSI internal RC oscillator.

1: FCLK and HCLK running, derived from the internal RC oscillator remaining active. If Systick is enabled, it may generate periodic interrupt and wake up events.

Upon Stop mode exit, the software must re-establish the desired clock configuration.

Bit 0 Reserved, must be kept at reset value.

### 26.10.3 DBG APB freeze register 1 (DBG\_APB\_FZ1)

Address offset: 0x08

Reset value: 0x0000 0000 (power-on reset)

Only 32-bit access are supported.

This register configures the clocking of timers, RTC, IWDG, WWDG, and I2C SMBUS peripherals of the MCU under debug:

The register is asynchronously reset by the POR but not affected by the system reset). It can be written by the debugger under 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.	Res.	Res.	DBG_I2C1_SMBUS_TIMEOUT	Res.	Res.	Res.	Res.	Res.
										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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw										rw

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

Bit 21 **DBG\_I2C1\_SMBUS\_TIMEOUT:** SMBUS timeout when core is halted

- 0: Same behavior as in normal mode
- 1: The SMBUS timeout is frozen

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG\_IWDG\_STOP:** Clocking of IWDG counter when the core is halted

- This bit enables/disables the clock to the counter of IWDG when the core is halted:
- 0: Enable
  - 1: Disable

Bit 11 **DBG\_WWDG\_STOP:** Clocking of WWDG counter when the core is halted

- This bit enables/disables the clock to the counter of WWDG when the core is halted:
- 0: Enable
  - 1: Disable

Bit 10 **DBG\_RTC\_STOP:** Clocking of RTC counter when the core is halted

- This bit enables/disables the clock to the counter of RTC when the core is halted:
- 0: Enable
  - 1: Disable

Bits 9:2 Reserved, must be kept at reset value.

Bit 1 **DBG\_TIM3\_STOP:** Clocking of TIM3 counter when the core is halted

- This bit enables/disables the clock to the counter of TIM3 when the core is halted:
- 0: Enable
  - 1: Disable

Bit 0 Reserved, must be kept at reset value.

#### 26.10.4 DBG APB freeze register 2 (DBG\_APB\_FZ2)

Address offset: 0x0C

Reset value: 0x0000 0000 (power-on reset)

Only 32-bit access is supported.

This register configures the clocking of timer counters when the MCU is under debug. It is asynchronously reset by the POR but not affected by the system reset. It can be written by the debugger under 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.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_TIM14_STOP	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.									
rw				rw											

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

Bit 18 **DBG\_TIM17\_STOP**: Clocking of TIM17 counter when the core is halted

This bit enables/disables the clock to the counter of TIM17 when the core is halted:

0: Enable

1: Disable

Bit 17 **DBG\_TIM16\_STOP**: Clocking of TIM16 counter when the core is halted

This bit enables/disables the clock to the counter of TIM16 when the core is halted:

0: Enable

1: Disable

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DBG\_TIM14\_STOP**: Clocking of TIM14 counter when the core is halted

This bit enables/disables the clock to the counter of TIM14 when the core is halted:

0: Enable

1: Disable

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

Bit 11 **DBG\_TIM1\_STOP**: Clocking of TIM1 counter when the core is halted

This bit enables/disables the clock to the counter of TIM1 when the core is halted:

0: Enable

1: Disable

Bits 10:0 Reserved, must be kept at reset value.

## 26.10.5 DBG register map

The following table summarizes the DBG registers.

**Table 127. DBG register map and reset values**

- The reset value is product dependent. For more information, refer to [Section 26.10.1: DBG device ID code register \(DBG\\_IDCODE\)](#).

Refer to [Section 2.2 on page 39](#) for the register boundary addresses.

## 27 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 STM32C0x1 microcontroller.

### 27.1 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: 0xFFFF 7550

Address offset: 0x00

Reset value: 0xXXXX XXXX where X is factory-programmed

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

Bits 31:0 **UID[31:0]**: X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Reset value: 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Reset value: 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]: LOT\_NUM[25:24]**

Lot number (ASCII encoded)

## 27.2 Flash memory size data register

Base address: 0x1FFF 75A0

Address offset: 0x00

Reset value: 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH\_SIZE[15:0]: Flash memory size**

This bitfield indicates the size of the device Flash memory expressed in Kbytes.

As an example, 0x040 corresponds to 64 Kbytes.

## 27.3 Package data register

Base address: 0x1FFF 7500

Address offset: 0x00

Reset value: 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.															
												r	r	r	r

Bits 15:4 Reserved

Bits 3:0 **PKG[3:0]**: Package type

**Condition: STM32C031xx**

0001: Reserved

0010: TSSOP20

0011: UFQFPN28

0100: UFQFPN32 / LQFP32

0101: UFQFPN48 / LQPF48

Others: Reserved

**Condition: STM32C011xx**

0001: SO8

0010: WLCSP12

0011: UFQFPN20

0100: TSSOP20

Others: Reserved

## 28      Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture ([www.psacertified.org](http://www.psacertified.org)) and/or Security Evaluation standard for IoT Platforms ([www.trustcb.com](http://www.trustcb.com)). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on [www.st.com](http://www.st.com) for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

## 29 Revision history

**Table 128. Document revision history**

Date	Revision	Changes
12-Apr-2022	1	Initial release.
21-Jul-2022	2	<p>Section <i>Fast programming</i> - row size corrected.</p> <p><i>Table 10: Organization of option bytes</i> now contains links to option registers and the duplicated description is removed.</p> <p>Format of reset values of option registers updated and/or corrected.</p> <p>Note in bit 16 of <i>FLASH security register (FLASH_SECR)</i> updated.</p> <p>Updated <i>Section 15.3.18: Clearing the OCxREF signal on an external event</i>.</p> <p>OC1M[3:0] bitfield description updated in <i>Section 15.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)</i>, <i>Section 16.4.8: TIM3 capture/compare mode register 1 [alternate] (TIM3_CCMR1)</i>, <i>Section 17.4.6: TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1)</i>, and <i>Section 18.4.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)</i> (<math>x = 16</math> to <math>17</math>).</p> <p>USART2 information in <i>Table 19: Device resources enabled in different operating modes</i> corrected.</p> <p>Spurious “TIM15” removed from <i>Section 5.2.6: Clock security system (CSS)</i>.</p> <p>Cross-reference to DBG added in WWDG <i>Section 21.3.5: Debug mode</i>.</p> <p><i>Section 22.2: RTC main features</i> corrected (spurious “or by a tamper event” removed).</p>
03-Dec-2022	3	<p>First public release.</p> <p><i>Section 4.3.2: Low-power modes</i>, bulleted point Standby mode.</p> <p><i>Section 5.2: Clocks</i>, bulleted point TIMx.</p> <p><i>Figure 9: Clock tree</i>.</p> <p><i>Section 5.3: Low-power modes</i>, removal of “USART2”.</p> <p>Section <i>Converting a supply-relative ADC measurement to an absolute voltage value</i>.</p> <p><i>Figure 98: Break and Break2 circuitry overview</i> - note under the figure corrected.</p> <p><i>Section 15.3.25: Interfacing with Hall sensors</i> - removal of “TIM4”.</p> <p><i>Figure 201: Break circuitry overview</i> - note under the figure corrected.</p>

# Index

## A

ADC_AWD1TR .....	287
ADC_AWD2CR .....	293
ADC_AWD2TR .....	288
ADC_AWD3CR .....	293
ADC_AWD3TR .....	292
ADC_CALFACT .....	294
ADC_CCR .....	294
ADC_CFGR1 .....	282
ADC_CFGR2 .....	285
ADC_CHSELR .....	289-290
ADC_CR .....	280
ADC_DR .....	292
ADC_IER .....	278
ADC_ISR .....	276
ADC_SMPR .....	286

## C

CRC_CR .....	235
CRC_DR .....	234
CRC_IDR .....	234
CRC_INIT .....	236
CRC_POL .....	236

## D

DBG_APB_FZ1 .....	813
DBG_APB_FZ2 .....	814
DBG_CR .....	813
DBG_IDCODE .....	812
DMA_CCRx .....	193
DMA_CMARx .....	197
DMA_CNDTRx .....	195
DMA_CPARx .....	196
DMA_IFCR .....	192
DMA_ISR .....	191
DMAMUX_CFR .....	210
DMAMUX_CSR .....	210
DMAMUX_CxCR .....	209
DMAMUX_RGCFR .....	212
DMAMUX_RGSR .....	212
DMAMUX_RGxCR .....	211

## E

EXTI_EMR1 .....	228
EXTI_EXTICRx .....	226

EXTI_FPR1 .....	225
EXTI_FTSR1 .....	224
EXTI_IMR1 .....	227
EXTI_RPR1 .....	225
EXTI_RTSR1 .....	223
EXTI_SWIER1 .....	224

## F

FLASH_ACR .....	67
FLASH_CR .....	70
FLASH_KEYR .....	68
FLASH_OPTKEYR .....	68
FLASH_OPTR .....	72
FLASH_PCROP1AER .....	75
FLASH_PCROP1ASR .....	75
FLASH_PCROP1BER .....	78
FLASH_PCROP1BSR .....	77
FLASH_SECR .....	78
FLASH_SR .....	69
FLASH_WRP1AR .....	76
FLASH_WRP1BR .....	77

## G

GPIOx_AFRH .....	159
GPIOx_AFRL .....	158
GPIOx_BRR .....	159
GPIOx_BSRR .....	156
GPIOx_IDR .....	156
GPIOx_LCKR .....	157
GPIOx_MODER .....	154
GPIOx_ODR .....	156
GPIOx_OSPEEDR .....	155
GPIOx_OTYPER .....	154
GPIOx_PUPDR .....	155

## I

I2C_CR1 .....	643
I2C_CR2 .....	646
I2C_ICR .....	654
I2C_ISR .....	651
I2C_OAR1 .....	648
I2C_OAR2 .....	648
I2C_PECR .....	655
I2C_RXDR .....	655
I2C_TIMEOUTR .....	650
I2C_TIMINGR .....	649

I2C_TXDR .....	656	RTC_CR .....	579
IWDG_KR .....	549	RTC_DR .....	576
IWDG_PR .....	550	RTC_ICSR .....	577
IWDG_RLR .....	551	RTC_MISR .....	588
IWDG_SR .....	552	RTC_PRER .....	579
IWDG_WINR .....	553	RTC_SCR .....	589
RTC_SHIFTR .....	583	RTC_SR .....	587
RTC_SSR .....	577	RTC_TR .....	575
RTC_TSDDR .....	585	RTC_TSSSR .....	585
RTC_TSTR .....	584	RTC_WPR .....	582
<b>P</b>		<b>S</b>	
PWR_BKPxR .....	101	SPIx_CR1 .....	792
PWR_CR1 .....	90	SPIx_CR2 .....	794
PWR_CR3 .....	91	SPIx_CRCPR .....	798
PWR_CR4 .....	92	SPIx_DR .....	798
PWR_PDCRA .....	96	SPIx_I2SCFGR .....	799
PWR_PDCRB .....	97	SPIx_I2SPR .....	801
PWR_PDCRC .....	98	SPIx_RXCRCR .....	798
PWR_PDCRD .....	99	SPIx_SR .....	796
PWR_PDCRF .....	100	SPIx_TXCRCR .....	799
PWR_PUCRA .....	95	SYSCFG_CFGR1 .....	161
PWR_PUCRB .....	97	SYSCFG_CFGR2 .....	163
PWR_PUCRC .....	98	SYSCFG_CFGR3 .....	164
PWR_PUCRD .....	99	SYSCFG_ITLINE0 .....	165
PWR_PUCRF .....	100	SYSCFG_ITLINE10 .....	169
PWR_SCR .....	95	SYSCFG_ITLINE11 .....	170
PWR_SR1 .....	93	SYSCFG_ITLINE12 .....	170
PWR_SR2 .....	94	SYSCFG_ITLINE13 .....	170
<b>R</b>		SYSCFG_ITLINE14 .....	171
RCC_AHBENR .....	130	SYSCFG_ITLINE16 .....	171
RCC_AHBRSTR .....	126	SYSCFG_ITLINE19 .....	171
RCC_AHBSMENR .....	134	SYSCFG_ITLINE2 .....	166
RCC_APBENR1 .....	130	SYSCFG_ITLINE21 .....	172
RCC_APBENR2 .....	132	SYSCFG_ITLINE22 .....	172
RCC_APBRSTR1 .....	126	SYSCFG_ITLINE23 .....	172
RCC_APBRSTR2 .....	127	SYSCFG_ITLINE25 .....	173
RCC_APBSMENR1 .....	134	SYSCFG_ITLINE27 .....	173
RCC_APBSMENR2 .....	136	SYSCFG_ITLINE28 .....	173
RCC_CCIPR .....	137	SYSCFG_ITLINE3 .....	166
RCC_CFGR .....	120	SYSCFG_ITLINE4 .....	167
RCC_CICR .....	124	SYSCFG_ITLINE5 .....	168
RCC_CIER .....	122	SYSCFG_ITLINE6 .....	168
RCC_CIFR .....	123	SYSCFG_ITLINE7 .....	168
RCC_CR .....	117	SYSCFG_ITLINE9 .....	169
RCC_CR2 .....	138		
RCC_CSR .....	140		
RCC_ICSCR .....	119		
RCC_IOPENR .....	129		
RCC_IOPRSTR .....	125		
RCC_IOPSMENR .....	133		
RTC_ALRMAR .....	586		
RTC_ALRMASSR .....	587		
RTC_CALR .....	582		

## Index

---

### T

TIM1_AF1 .....	390
TIM1_AF2 .....	391
TIM1_ARR .....	379
TIM1_BDTR .....	382
TIM1_CCER .....	376
TIM1_CCMR1 .....	369-370
TIM1_CCMR2 .....	373-374
TIM1_CCMR3 .....	388
TIM1_CCR1 .....	380
TIM1_CCR2 .....	381
TIM1_CCR3 .....	381
TIM1_CCR4 .....	382
TIM1_CCR5 .....	389
TIM1_CCR6 .....	390
TIM1_CNT .....	379
TIM1_CR1 .....	358
TIM1_CR2 .....	359
TIM1_DCR .....	386
TIM1_DIER .....	364
TIM1_DMAR .....	387
TIM1_EGR .....	368
TIM1_PSC .....	379
TIM1_RCR .....	380
TIM1_SMCR .....	362
TIM1_SR .....	366
TIM1_TISEL .....	392
TIM14_ARR .....	490
TIM14_CCER .....	488
TIM14_CCMR1 .....	485-486
TIM14_CCR1 .....	490
TIM14_CNT .....	489
TIM14_CR1 .....	482
TIM14_DIER .....	483
TIM14_EGR .....	484
TIM14_PSC .....	490
TIM14_SR .....	483
TIM14_TISEL .....	491
TIM16_AF1 .....	541
TIM16_TISEL .....	541
TIM17_AF1 .....	542
TIM17_TISEL .....	542
TIM3_AF1 .....	464
TIM3_ARR .....	460
TIM3_CCER .....	457
TIM3_CCMR1 .....	451, 453
TIM3_CCMR2 .....	455-456
TIM3_CCR1 .....	460
TIM3_CCR2 .....	461
TIM3_CCR3 .....	461
TIM3_CCR4 .....	462
TIM3_CNT .....	458-459

TIM3_CR1 .....	441
TIM3_CR2 .....	442
TIM3_DCR .....	463
TIM3_DIER .....	447
TIM3_DMAR .....	463
TIM3_EGR .....	450
TIM3_PSC .....	459
TIM3_SMCR .....	444
TIM3_SR .....	448
TIM3_TISEL .....	464
TIMx_ARR .....	535
TIMx_BDTR .....	537
TIMx_CCER .....	532
TIMx_CCMR1 .....	529-530
TIMx_CCR1 .....	536
TIMx_CNT .....	534
TIMx_CR1 .....	524
TIMx_CR2 .....	525
TIMx_DCR .....	540
TIMx_DIER .....	526
TIMx_DMAR .....	540
TIMx_EGR .....	528
TIMx_PSC .....	535
TIMx_RCR .....	536
TIMx_SR .....	527

### U

USART_BRR .....	726
USART_CR1 .....	710, 714
USART_CR2 .....	717
USART_CR3 .....	721
USART_GTPR .....	726
USART_ICR .....	740
USART_ISR .....	729, 735
USART_PRESC .....	743
USART_RDR .....	742
USART_RQR .....	728
USART_RTOR .....	727
USART_TDR .....	742

### W

WWDG_CFR .....	559
WWDG_CR .....	558
WWDG_SR .....	560

**IMPORTANT NOTICE – READ CAREFULLY**

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2022 STMicroelectronics – All rights reserved