

## Introduction

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

The devices include ST state-of-the-art patented technology.

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

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

## Related documents

- “Cortex<sup>®</sup>-M0+ Technical Reference Manual”, available from: <http://infocenter.arm.com>
- PM0223 programming manual for Cortex<sup>®</sup>-M0+ core<sup>(a)</sup>
- STM32G0x1 datasheets<sup>(a)</sup>
- AN2606 application note on booting STM32 MCUs<sup>(a)</sup>
- STM32G0x1 device errata sheets<sup>(a)</sup>

---

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

# Contents

<b>1</b>	<b>Documentation conventions</b>	<b>55</b>
1.1	General information	55
1.2	List of abbreviations for registers	55
1.3	Glossary	56
1.4	Availability of peripherals	56
<b>2</b>	<b>Memory and bus architecture</b>	<b>58</b>
2.1	System architecture	58
2.2	Memory organization	60
2.2.1	Introduction	60
2.2.2	Memory map and register boundary addresses	61
2.3	Embedded SRAM	66
2.4	Flash memory overview	67
2.5	Boot configuration	68
2.5.1	Physical remap	69
2.5.2	Embedded bootloader	69
2.5.3	Forcing boot from main flash memory	69
2.5.4	Empty check	69
<b>3</b>	<b>Embedded flash memory (FLASH)</b>	<b>70</b>
3.1	FLASH Introduction	70
3.2	FLASH main features	70
3.3	FLASH functional description	71
3.3.1	FLASH organization	71
3.3.2	FLASH dual-bank capability	73
3.3.3	FLASH error code correction (ECC)	74
3.3.4	FLASH read access latency	75
3.3.5	FLASH memory acceleration	76
3.3.6	FLASH program and erase operations	77
3.3.7	FLASH main memory erase sequences	77
3.3.8	FLASH main memory programming sequences	79
3.3.9	Read-while-write (RWW) function	82
3.4	FLASH option bytes	82

3.4.1	FLASH option byte description	82
3.4.2	FLASH option byte programming	84
3.5	FLASH memory protection	85
3.5.1	FLASH read protection (RDP)	86
3.5.2	FLASH proprietary code readout protection (PCROP)	88
3.5.3	FLASH write protection (WRP)	90
3.5.4	Securable memory area	91
3.5.5	Disabling core debug access	92
3.5.6	Forcing boot from main flash memory	92
3.6	FLASH interrupts	93
3.7	FLASH registers	94
3.7.1	FLASH access control register (FLASH_ACR)	94
3.7.2	FLASH key register (FLASH_KEYR)	95
3.7.3	FLASH option key register (FLASH_OPTKEYR)	95
3.7.4	FLASH status register (FLASH_SR)	96
3.7.5	FLASH control register (FLASH_CR)	98
3.7.6	FLASH ECC register (FLASH_ECCR)	100
3.7.7	FLASH ECC register 2 (FLASH_ECCR2)	101
3.7.8	FLASH option register (FLASH_OPTR)	102
3.7.9	FLASH PCROP area A start address register (FLASH_PCROP1ASR)	104
3.7.10	FLASH PCROP area A end address register (FLASH_PCROP1AER)	105
3.7.11	FLASH WRP area A address register (FLASH_WRP1AR)	105
3.7.12	FLASH WRP area B address register (FLASH_WRP1BR)	106
3.7.13	FLASH PCROP area B start address register (FLASH_PCROP1BSR)	107
3.7.14	FLASH PCROP area B end address register (FLASH_PCROP1BER)	107
3.7.15	FLASH PCROP2 area A start address register (FLASH_PCROP2ASR)	108
3.7.16	FLASH PCROP2 area A end address register (FLASH_PCROP2AER)	108
3.7.17	FLASH WRP2 area A address register (FLASH_WRP2AR)	109
3.7.18	FLASH WRP2 area B address register (FLASH_WRP2BR)	109
3.7.19	FLASH PCROP2 area B start address register (FLASH_PCROP2BSR)	110
3.7.20	FLASH PCROP2 area B end address register (FLASH_PCROP2BER)	110

3.7.21	FLASH security register (FLASH_SECR) .....	111
3.7.22	FLASH register map .....	112
<b>4</b>	<b>Power control (PWR) .....</b>	<b>114</b>
4.1	Power supplies .....	114
4.1.1	ADC and DAC reference voltage .....	115
4.1.2	Battery backup of RTC domain .....	115
4.1.3	Voltage regulator .....	117
4.1.4	Dynamic voltage scaling management .....	118
4.2	Power supply supervisor .....	119
4.2.1	Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR) .....	119
4.2.2	Programmable voltage detector (PVD) .....	120
4.3	Low-power modes .....	121
4.3.1	Run mode .....	126
4.3.2	Low-power run mode (LP run) .....	126
4.3.3	Low-power modes .....	127
4.3.4	Sleep mode .....	128
4.3.5	Low-power sleep mode (LP sleep) .....	128
4.3.6	Stop 0 mode .....	129
4.3.7	Stop 1 mode .....	132
4.3.8	Standby mode .....	133
4.3.9	Shutdown mode .....	135
4.3.10	Auto-wake-up from low-power mode .....	136
4.4	PWR registers .....	137
4.4.1	Power control register 1 (PWR_CR1) .....	137
4.4.2	Power control register 2 (PWR_CR2) .....	138
4.4.3	Power control register 3 (PWR_CR3) .....	140
4.4.4	Power control register 4 (PWR_CR4) .....	141
4.4.5	Power status register 1 (PWR_SR1) .....	142
4.4.6	Power status register 2 (PWR_SR2) .....	143
4.4.7	Power status clear register (PWR_SCR) .....	145
4.4.8	Power Port A pull-up control register (PWR_PUCRA) .....	146
4.4.9	Power Port A pull-down control register (PWR_PDCRA) .....	146
4.4.10	Power Port B pull-up control register (PWR_PUCRB) .....	147
4.4.11	Power Port B pull-down control register (PWR_PDCRB) .....	147
4.4.12	Power Port C pull-up control register (PWR_PUCRC) .....	148



4.4.13	Power Port C pull-down control register (PWR_PDCRC)	148
4.4.14	Power Port D pull-up control register (PWR_PUCRD)	149
4.4.15	Power Port D pull-down control register (PWR_PDCRD)	149
4.4.16	Power Port E pull-up control register (PWR_PUCRE)	150
4.4.17	Power Port E pull-down control register (PWR_PDCRE)	150
4.4.18	Power Port F pull-up control register (PWR_PUCRF)	151
4.4.19	Power Port F pull-down control register (PWR_PDCRF)	151
4.4.20	PWR register map	153
<b>5</b>	<b>Reset and clock control (RCC)</b>	<b>155</b>
5.1	Reset	155
5.1.1	Power reset	155
5.1.2	System reset	155
5.1.3	RTC domain reset	157
5.2	Clocks	158
5.2.1	HSE clock	162
5.2.2	HSI16 clock	163
5.2.3	HSI48 clock	164
5.2.4	PLL	164
5.2.5	LSE clock	165
5.2.6	LSI clock	165
5.2.7	System clock (SYSCLK) selection	166
5.2.8	Clock source frequency versus voltage scaling	166
5.2.9	Clock security system (CSS)	166
5.2.10	Clock security system for LSE clock (LSECSS)	167
5.2.11	ADC clock	167
5.2.12	RTC clock	168
5.2.13	Timer clock	168
5.2.14	Watchdog clock	168
5.2.15	Clock-out capability	169
5.2.16	Internal/external clock measurement with TIM14/TIM16/TIM17	169
5.2.17	Peripheral clock enable registers	172
5.3	Low-power modes	172
5.4	RCC registers	174
5.4.1	Clock control register (RCC_CR)	174
5.4.2	Internal clock source calibration register (RCC_ICSCR)	176
5.4.3	Clock configuration register (RCC_CFGR)	176

5.4.4	PLL configuration register (RCC_PLLCFGR) .....	179
5.4.5	RCC clock recovery RC register (RCC_CRRCR) .....	182
5.4.6	Clock interrupt enable register (RCC_CIER) .....	182
5.4.7	Clock interrupt flag register (RCC_CIFR) .....	183
5.4.8	Clock interrupt clear register (RCC_CICR) .....	185
5.4.9	I/O port reset register (RCC_IOPRSTR) .....	186
5.4.10	AHB peripheral reset register (RCC_AHBRSTR) .....	187
5.4.11	APB peripheral reset register 1 (RCC_APBRSTR1) .....	188
5.4.12	APB peripheral reset register 2 (RCC_APBRSTR2) .....	191
5.4.13	I/O port clock enable register (RCC_IOPENR) .....	192
5.4.14	AHB peripheral clock enable register (RCC_AHBENR) .....	193
5.4.15	APB peripheral clock enable register 1 (RCC_APBENR1) .....	194
5.4.16	APB peripheral clock enable register 2 (RCC_APBENR2) .....	198
5.4.17	I/O port in Sleep mode clock enable register (RCC_IOPSMENR) ...	199
5.4.18	AHB peripheral clock enable in Sleep/Stop mode register (RCC_AHBSMENR) .....	200
5.4.19	APB peripheral clock enable in Sleep/Stop mode register 1 (RCC_APBSMENR1) .....	201
5.4.20	APB peripheral clock enable in Sleep/Stop mode register 2 (RCC_APBSMENR2) .....	205
5.4.21	Peripherals independent clock configuration register (RCC_CCIPR) .	206
5.4.22	Peripherals independent clock configuration register 2 (RCC_CCIPR2) .....	208
5.4.23	RTC domain control register (RCC_BDCR) .....	209
5.4.24	Control/status register (RCC_CSR) .....	211
5.4.25	RCC register map .....	213
<b>6</b>	<b>Clock recovery system (CRS) .....</b>	<b>217</b>
6.1	CRS introduction .....	217
6.2	CRS main features .....	217
6.3	CRS implementation .....	217
6.4	CRS functional description .....	218
6.4.1	CRS block diagram .....	218
6.4.2	CRS internal signals .....	218
6.4.3	Synchronization input .....	219
6.4.4	Frequency error measurement .....	219
6.4.5	Frequency error evaluation and automatic trimming .....	220
6.4.6	CRS initialization and configuration .....	221

6.5	CRS in low-power modes	222
6.6	CRS interrupts	222
6.7	CRS registers	222
6.7.1	CRS control register (CRS_CR)	222
6.7.2	CRS configuration register (CRS_CFGR)	223
6.7.3	CRS interrupt and status register (CRS_ISR)	224
6.7.4	CRS interrupt flag clear register (CRS_ICR)	226
6.7.5	CRS register map	227
<b>7</b>	<b>General-purpose I/Os (GPIO)</b>	<b>228</b>
7.1	Introduction	228
7.2	GPIO main features	228
7.3	GPIO functional description	228
7.3.1	General-purpose I/O (GPIO)	230
7.3.2	I/O pin alternate function multiplexer and mapping	230
7.3.3	I/O port control registers	232
7.3.4	I/O port state in low-power modes	232
7.3.5	I/O port data registers	232
7.3.6	I/O data bitwise handling	232
7.3.7	GPIO locking mechanism	233
7.3.8	I/O alternate function input/output	233
7.3.9	External interrupt/wake-up lines	233
7.3.10	Input configuration	233
7.3.11	Output configuration	234
7.3.12	Alternate function configuration	235
7.3.13	Analog configuration	236
7.3.14	Using the HSE or LSE oscillator pins as GPIOs	237
7.3.15	Using the GPIO pins in the RTC domain	237
7.3.16	USB PD / Dead battery support	237
7.3.17	Reset pin (PF2-NRST) in GPIO mode	237
7.4	GPIO in low-power modes	238
7.5	GPIO registers	238
7.5.1	GPIO port mode register (GPIOx_MODER) (x = A to F)	238
7.5.2	GPIO port output type register (GPIOx_OTYPER) (x = A to F)	239

- 7.5.3 GPIO port output speed register (GPIOx\_OSPEEDR)  
(x = A to F) ..... 239
- 7.5.4 GPIO port pull-up/pull-down register (GPIOx\_PUPDR)  
(x = A to F) ..... 240
- 7.5.5 GPIO port input data register (GPIOx\_IDR)  
(x = A to F) ..... 240
- 7.5.6 GPIO port output data register (GPIOx\_ODR)  
(x = A to F) ..... 241
- 7.5.7 GPIO port bit set/reset register (GPIOx\_BSRR)  
(x = A to F) ..... 241
- 7.5.8 GPIO port configuration lock register (GPIOx\_LCKR)  
(x = A to F) ..... 241
- 7.5.9 GPIO port alternate function low register (GPIOx\_AFRL)  
(x = A to F) ..... 242
- 7.5.10 GPIO port alternate function high register (GPIOx\_AFRH)  
(x = A to F) ..... 243
- 7.5.11 GPIO port bit reset register (GPIOx\_BRR) (x = A to F) ..... 243
- 7.5.12 GPIO register map ..... 245

**8 System configuration controller (SYSCFG) ..... 246**

- 8.1 SYSCFG registers ..... 246
  - 8.1.1 SYSCFG configuration register 1 (SYSCFG\_CFGR1) ..... 246
  - 8.1.2 SYSCFG configuration register 2 (SYSCFG\_CFGR2) ..... 249
  - 8.1.3 SYSCFG interrupt line 0 status register (SYSCFG\_ITLINE0) ..... 251
  - 8.1.4 SYSCFG interrupt line 1 status register (SYSCFG\_ITLINE1) ..... 252
  - 8.1.5 SYSCFG interrupt line 2 status register (SYSCFG\_ITLINE2) ..... 252
  - 8.1.6 SYSCFG interrupt line 3 status register (SYSCFG\_ITLINE3) ..... 252
  - 8.1.7 SYSCFG interrupt line 4 status register (SYSCFG\_ITLINE4) ..... 253
  - 8.1.8 SYSCFG interrupt line 5 status register (SYSCFG\_ITLINE5) ..... 254
  - 8.1.9 SYSCFG interrupt line 6 status register (SYSCFG\_ITLINE6) ..... 254
  - 8.1.10 SYSCFG interrupt line 7 status register (SYSCFG\_ITLINE7) ..... 254
  - 8.1.11 SYSCFG interrupt line 8 status register (SYSCFG\_ITLINE8) ..... 255
  - 8.1.12 SYSCFG interrupt line 9 status register (SYSCFG\_ITLINE9) ..... 255
  - 8.1.13 SYSCFG interrupt line 10 status register (SYSCFG\_ITLINE10) ..... 256
  - 8.1.14 SYSCFG interrupt line 11 status register (SYSCFG\_ITLINE11) ..... 256
  - 8.1.15 SYSCFG interrupt line 12 status register (SYSCFG\_ITLINE12) ..... 257
  - 8.1.16 SYSCFG interrupt line 13 status register (SYSCFG\_ITLINE13) ..... 257
  - 8.1.17 SYSCFG interrupt line 14 status register (SYSCFG\_ITLINE14) ..... 258
  - 8.1.18 SYSCFG interrupt line 15 status register (SYSCFG\_ITLINE15) ..... 258



8.1.19	SYSCFG interrupt line 16 status register (SYSCFG_ITLINE16) . . . . .	258
8.1.20	SYSCFG interrupt line 17 status register (SYSCFG_ITLINE17) . . . . .	259
8.1.21	SYSCFG interrupt line 18 status register (SYSCFG_ITLINE18) . . . . .	259
8.1.22	SYSCFG interrupt line 19 status register (SYSCFG_ITLINE19) . . . . .	260
8.1.23	SYSCFG interrupt line 20 status register (SYSCFG_ITLINE20) . . . . .	260
8.1.24	SYSCFG interrupt line 21 status register (SYSCFG_ITLINE21) . . . . .	260
8.1.25	SYSCFG interrupt line 22 status register (SYSCFG_ITLINE22) . . . . .	261
8.1.26	SYSCFG interrupt line 23 status register (SYSCFG_ITLINE23) . . . . .	261
8.1.27	SYSCFG interrupt line 24 status register (SYSCFG_ITLINE24) . . . . .	262
8.1.28	SYSCFG interrupt line 25 status register (SYSCFG_ITLINE25) . . . . .	262
8.1.29	SYSCFG interrupt line 26 status register (SYSCFG_ITLINE26) . . . . .	262
8.1.30	SYSCFG interrupt line 27 status register (SYSCFG_ITLINE27) . . . . .	263
8.1.31	SYSCFG interrupt line 28 status register (SYSCFG_ITLINE28) . . . . .	263
8.1.32	SYSCFG interrupt line 29 status register (SYSCFG_ITLINE29) . . . . .	264
8.1.33	SYSCFG interrupt line 30 status register (SYSCFG_ITLINE30) . . . . .	264
8.1.34	SYSCFG interrupt line 31 status register (SYSCFG_ITLINE31) . . . . .	264
8.1.35	SYSCFG register map . . . . .	265
<b>9</b>	<b>Interconnect matrix . . . . .</b>	<b>268</b>
9.1	Introduction . . . . .	268
9.2	Connection summary . . . . .	268
9.3	Interconnection details . . . . .	269
9.3.1	From TIM1, TIM2, TIM3, TIM4, TIM15, TIM16, and TIM17, to TIM1, TIM2, TIM3, TIM4, and TIM15 . . . . .	269
9.3.2	From TIM1, TIM2, TIM3, TIM4, TIM6, TIM15, and EXTI, to ADC . . . . .	270
9.3.3	From ADC to TIM1 . . . . .	270
9.3.4	From TIM1, TIM2, TIM3, TIM4, TIM6, TIM7, TIM15, LPTIM1, LPTIM2, and EXTI, to DAC . . . . .	271
9.3.5	From HSE, LSE, LSI, MCO, MCO2, RTC and TAMP, to TIM2, TIM14, TIM16, and TIM17 . . . . .	271
9.3.6	From RTC, TAMP, COMP1, COMP2, and COMP3 to LPTIM1 and LPTIM2 . . . . .	272
9.3.7	From TIM1, TIM2, TIM3, TIM4, and TIM15, to COMP1, COMP2, and COMP3 . . . . .	272
9.3.8	From internal analog sources to ADC . . . . .	272
9.3.9	From COMP1, COMP2, and COMP3 to TIM1, TIM2, TIM3, TIM4, TIM15, TIM16, and TIM17 . . . . .	273
9.3.10	From system errors to TIM1, TIM2, TIM3, TIM4, TIM15, TIM16, and TIM17 . . . . .	273

9.3.11 From TIM16, TIM17, USART1, and USART4, to IRTIM ..... 274

9.3.12 From TIM14, LPTIM1, and LPTIM2, to DMAMUX ..... 274

**10 Direct memory access controller (DMA) ..... 275**

10.1 Introduction ..... 275

10.2 DMA main features ..... 275

10.3 DMA implementation ..... 276

10.3.1 DMA ..... 276

10.3.2 DMA request mapping ..... 276

10.4 DMA functional description ..... 277

10.4.1 DMA block diagram ..... 277

10.4.2 DMA pins and internal signals ..... 277

10.4.3 DMA transfers ..... 278

10.4.4 DMA arbitration ..... 279

10.4.5 DMA channels ..... 279

10.4.6 DMA data width, alignment, and endianness ..... 283

10.4.7 DMA error management ..... 284

10.5 DMA interrupts ..... 285

10.6 DMA registers ..... 285

10.6.1 DMA interrupt status register (DMA\_ISR) ..... 285

10.6.2 DMA interrupt flag clear register (DMA\_IFCR) ..... 287

10.6.3 DMA channel x configuration register (DMA\_CCRx) ..... 289

10.6.4 DMA channel x number of data to transfer register (DMA\_CNDTRx) . 291

10.6.5 DMA channel x peripheral address register (DMA\_CPARx) ..... 292

10.6.6 DMA channel x memory address register (DMA\_CMARx) ..... 293

10.6.7 DMA register map ..... 293

**11 DMA request multiplexer (DMAMUX) ..... 296**

11.1 Introduction ..... 296

11.2 DMAMUX main features ..... 297

11.3 DMAMUX implementation ..... 297

11.3.1 DMAMUX instantiation ..... 297

11.3.2 DMAMUX mapping ..... 297

11.4 DMAMUX functional description ..... 300

11.4.1 DMAMUX block diagram ..... 300

11.4.2 DMAMUX signals ..... 301

11.4.3	DMAMUX channels .....	301
11.4.4	DMAMUX request line multiplexer .....	301
11.4.5	DMAMUX request generator .....	304
11.5	DMAMUX interrupts .....	305
11.6	DMAMUX registers .....	306
11.6.1	DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR) .....	306
11.6.2	DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR) .....	307
11.6.3	DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR) .....	307
11.6.4	DMAMUX request generator channel x configuration register (DMAMUX_RGxCR) .....	308
11.6.5	DMAMUX request generator interrupt status register (DMAMUX_RGSR) .....	309
11.6.6	DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR) .....	309
11.6.7	DMAMUX register map .....	310
<b>12</b>	<b>Nested vectored interrupt controller (NVIC) .....</b>	<b>312</b>
12.1	Main features .....	312
12.2	SysTick calibration value register .....	312
12.3	Interrupt and exception vectors .....	312
<b>13</b>	<b>Extended interrupt and event controller (EXTI) .....</b>	<b>315</b>
13.1	EXTI main features .....	315
13.2	EXTI block diagram .....	315
13.2.1	EXTI connections between peripherals and CPU .....	317
13.3	EXTI functional description .....	317
13.3.1	EXTI configurable event input wake-up .....	318
13.3.2	EXTI direct event input wake-up .....	319
13.3.3	EXTI mux .....	319
13.4	EXTI functional behavior .....	321
13.5	EXTI registers .....	322
13.5.1	EXTI rising trigger selection register (EXTI_RTISR1) .....	322
13.5.2	EXTI falling trigger selection register 1 (EXTI_FTSR1) .....	323
13.5.3	EXTI software interrupt event register 1 (EXTI_SWIER1) .....	323
13.5.4	EXTI rising edge pending register 1 (EXTI_RPR1) .....	324

13.5.5	EXTI falling edge pending register 1 (EXTI_FPR1)	325
13.5.6	EXTI rising trigger selection register 2 (EXTI_RTSTR2)	325
13.5.7	EXTI falling trigger selection register 2 (EXTI_FTSTR2)	326
13.5.8	EXTI software interrupt event register 2 (EXTI_SWIER2)	326
13.5.9	EXTI rising edge pending register 2 (EXTI_RPR2)	327
13.5.10	EXTI falling edge pending register 2 (EXTI_FPR2)	327
13.5.11	EXTI external interrupt selection register (EXTI_EXTICRx)	328
13.5.12	EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)	329
13.5.13	EXTI CPU wake-up with event mask register (EXTI_EMR1)	330
13.5.14	EXTI CPU wake-up with interrupt mask register (EXTI_IMR2)	330
13.5.15	EXTI CPU wake-up with event mask register (EXTI_EMR2)	331
13.5.16	EXTI register map	332
<b>14</b>	<b>Cyclic redundancy check calculation unit (CRC)</b>	<b>334</b>
14.1	Introduction	334
14.2	CRC main features	334
14.3	CRC functional description	335
14.3.1	CRC block diagram	335
14.3.2	CRC internal signals	335
14.3.3	CRC operation	335
14.4	CRC registers	337
14.4.1	CRC data register (CRC_DR)	337
14.4.2	CRC independent data register (CRC_IDR)	337
14.4.3	CRC control register (CRC_CR)	338
14.4.4	CRC initial value (CRC_INIT)	339
14.4.5	CRC polynomial (CRC_POL)	339
14.4.6	CRC register map	340
<b>15</b>	<b>Analog-to-digital converter (ADC)</b>	<b>341</b>
15.1	Introduction	341
15.2	ADC main features	342
15.3	ADC functional description	343
15.3.1	ADC pins and internal signals	343
15.3.2	ADC voltage regulator (ADVREGEN)	344
15.3.3	Calibration (ADCAL)	345
15.3.4	ADC on-off control (ADEN, ADDIS, ADRDY)	347



15.3.5	ADC clock (CKMODE, PRESC[3:0])	349
15.3.6	ADC connectivity	351
15.3.7	Configuring the ADC	352
15.3.8	Channel selection (CHSEL, SCANDIR, CHSELRMOD)	352
15.3.9	Programmable sampling time (SMPx[2:0])	353
15.3.10	Single conversion mode (CONT = 0)	354
15.3.11	Continuous conversion mode (CONT = 1)	354
15.3.12	Starting conversions (ADSTART)	355
15.3.13	Timings	356
15.3.14	Stopping an ongoing conversion (ADSTP)	357
15.4	Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)	357
15.4.1	Discontinuous mode (DISCEN)	358
15.4.2	Programmable resolution (RES) - Fast conversion mode	358
15.4.3	End of conversion, end of sampling phase (EOC, EOSMP flags)	359
15.4.4	End of conversion sequence (EOS flag)	359
15.4.5	Example timing diagrams (single/continuous modes hardware/software triggers)	360
15.4.6	Low frequency trigger mode	362
15.5	Data management	362
15.5.1	Data register and data alignment (ADC_DR, ALIGN)	362
15.5.2	ADC overrun (OVR, OVRMOD)	362
15.5.3	Managing a sequence of data converted without using the DMA	364
15.5.4	Managing converted data without using the DMA without overrun	364
15.5.5	Managing converted data using the DMA	364
15.6	Low-power features	365
15.6.1	Wait mode conversion	365
15.6.2	Auto-off mode (AUTOFF)	366
15.7	Analog window watchdogs	367
15.7.1	Description of analog watchdog 1	368
15.7.2	Description of analog watchdog 2 and 3	369
15.7.3	ADC_AWDx_OUT output signal generation	369
15.7.4	Analog watchdog threshold control	371
15.8	Oversampler	372
15.8.1	ADC operating modes supported when oversampling	373
15.8.2	Analog watchdog	374
15.8.3	Triggered mode	374
15.9	Temperature sensor and internal reference voltage	374

15.10	Battery voltage monitoring	377
15.11	ADC interrupts	378
15.12	ADC registers	379
15.12.1	ADC interrupt and status register (ADC_ISR)	379
15.12.2	ADC interrupt enable register (ADC_IER)	380
15.12.3	ADC control register (ADC_CR)	382
15.12.4	ADC configuration register 1 (ADC_CFGR1)	384
15.12.5	ADC configuration register 2 (ADC_CFGR2)	387
15.12.6	ADC sampling time register (ADC_SMPR)	389
15.12.7	ADC watchdog threshold register (ADC_AWD1TR)	390
15.12.8	ADC watchdog threshold register (ADC_AWD2TR)	390
15.12.9	ADC channel selection register (ADC_CHSELR)	390
15.12.10	ADC channel selection register [alternate] (ADC_CHSELR)	391
15.12.11	ADC watchdog threshold register (ADC_AWD3TR)	393
15.12.12	ADC data register (ADC_DR)	394
15.12.13	ADC analog watchdog 2 configuration register (ADC_AWD2CR)	394
15.12.14	ADC Analog Watchdog 3 Configuration register (ADC_AWD3CR)	395
15.12.15	ADC calibration factor (ADC_CALFACT)	395
15.12.16	ADC common configuration register (ADC_CCR)	396
15.13	ADC register map	397
<b>16</b>	<b>Digital-to-analog converter (DAC)</b>	<b>400</b>
16.1	Introduction	400
16.2	DAC main features	400
16.3	DAC implementation	401
16.4	DAC functional description	402
16.4.1	DAC block diagram	402
16.4.2	DAC pins and internal signals	403
16.4.3	DAC channel enable	404
16.4.4	DAC data format	404
16.4.5	DAC conversion	406
16.4.6	DAC output voltage	406
16.4.7	DAC trigger selection	406
16.4.8	DMA requests	407
16.4.9	Noise generation	407
16.4.10	Triangle-wave generation	409

16.4.11	DAC channel modes	410
16.4.12	DAC channel buffer calibration	413
16.4.13	DAC channel conversion modes	414
16.4.14	Dual DAC channel conversion modes (if dual channels are available)	415
16.5	DAC in low-power modes	419
16.6	DAC interrupts	420
16.7	DAC registers	420
16.7.1	DAC control register (DAC_CR)	420
16.7.2	DAC software trigger register (DAC_SWTRGR)	423
16.7.3	DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)	424
16.7.4	DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)	424
16.7.5	DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)	425
16.7.6	DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)	425
16.7.7	DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)	426
16.7.8	DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)	426
16.7.9	Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)	427
16.7.10	Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)	427
16.7.11	Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)	428
16.7.12	DAC channel1 data output register (DAC_DOR1)	428
16.7.13	DAC channel2 data output register (DAC_DOR2)	429
16.7.14	DAC status register (DAC_SR)	429
16.7.15	DAC calibration control register (DAC_CCR)	431
16.7.16	DAC mode control register (DAC_MCR)	431
16.7.17	DAC channel1 sample and hold sample time register (DAC_SHSR1)	433
16.7.18	DAC channel2 sample and hold sample time register (DAC_SHSR2)	433
16.7.19	DAC sample and hold time register (DAC_SHHR)	433
16.7.20	DAC sample and hold refresh time register (DAC_SHRR)	434
16.7.21	DAC register map	435

<b>17</b>	<b>Voltage reference buffer (VREFBUF) .....</b>	<b>437</b>
17.1	Introduction .....	437
17.2	VREFBUF functional description .....	437
17.3	VREFBUF registers .....	438
17.3.1	VREFBUF control and status register (VREFBUF_CSR) .....	438
17.3.2	VREFBUF calibration control register (VREFBUF_CCR) .....	438
17.3.3	VREFBUF register map .....	439
<b>18</b>	<b>Comparator (COMP) .....</b>	<b>440</b>
18.1	Introduction .....	440
18.2	COMP main features .....	440
18.3	COMP functional description .....	441
18.3.1	COMP block diagram .....	441
18.3.2	COMP pins and internal signals .....	441
18.3.3	COMP reset and clocks .....	443
18.3.4	Comparator LOCK mechanism .....	443
18.3.5	Window comparator .....	444
18.3.6	Hysteresis .....	444
18.3.7	Comparator output blanking function .....	445
18.3.8	COMP power and speed modes .....	445
18.4	COMP low-power modes .....	446
18.5	COMP interrupts .....	446
18.6	COMP registers .....	446
18.6.1	Comparator 1 control and status register (COMP1_CSR) .....	446
18.6.2	Comparator 2 control and status register (COMP2_CSR) .....	448
18.6.3	Comparator 3 control and status register (COMP3_CSR) .....	450
18.6.4	COMP register map .....	453
<b>19</b>	<b>True random number generator (RNG) .....</b>	<b>454</b>
19.1	Introduction .....	454
19.2	RNG main features .....	454
19.3	RNG functional description .....	455
19.3.1	RNG block diagram .....	455
19.3.2	RNG internal signals .....	455
19.3.3	Random number generation .....	456
19.3.4	RNG initialization .....	458

19.3.5	RNG operation	459
19.3.6	RNG clocking	460
19.3.7	Error management	460
19.3.8	RNG low-power use	461
19.4	RNG interrupts	461
19.5	RNG processing time	461
19.6	RNG entropy source validation	462
19.6.1	Introduction	462
19.6.2	Validation conditions	462
19.6.3	Data collection	462
19.7	RNG registers	463
19.7.1	RNG control register (RNG_CR)	463
19.7.2	RNG status register (RNG_SR)	464
19.7.3	RNG data register (RNG_DR)	465
19.7.4	RNG register map	465
<b>20</b>	<b>AES hardware accelerator (AES)</b>	<b>466</b>
20.1	Introduction	466
20.2	AES main features	466
20.3	AES implementation	466
20.4	AES functional description	467
20.4.1	AES block diagram	467
20.4.2	AES internal signals	467
20.4.3	AES cryptographic core	467
20.4.4	AES procedure to perform a cipher operation	473
20.4.5	AES decryption round key preparation	476
20.4.6	AES ciphertext stealing and data padding	476
20.4.7	AES task suspend and resume	477
20.4.8	AES basic chaining modes (ECB, CBC)	477
20.4.9	AES counter (CTR) mode	482
20.4.10	AES Galois/counter mode (GCM)	485
20.4.11	AES Galois message authentication code (GMAC)	490
20.4.12	AES counter with CBC-MAC (CCM)	492
20.4.13	AES data registers and data swapping	497
20.4.14	AES key registers	499
20.4.15	AES initialization vector registers	499

20.4.16	AES DMA interface	500
20.4.17	AES error management	501
20.5	AES interrupts	502
20.6	AES processing latency	502
20.7	AES registers	503
20.7.1	AES control register (AES_CR)	503
20.7.2	AES status register (AES_SR)	506
20.7.3	AES data input register (AES_DINR)	507
20.7.4	AES data output register (AES_DOUTR)	507
20.7.5	AES key register 0 (AES_KEYR0)	508
20.7.6	AES key register 1 (AES_KEYR1)	508
20.7.7	AES key register 2 (AES_KEYR2)	509
20.7.8	AES key register 3 (AES_KEYR3)	509
20.7.9	AES initialization vector register 0 (AES_IVR0)	509
20.7.10	AES initialization vector register 1 (AES_IVR1)	510
20.7.11	AES initialization vector register 2 (AES_IVR2)	510
20.7.12	AES initialization vector register 3 (AES_IVR3)	510
20.7.13	AES key register 4 (AES_KEYR4)	511
20.7.14	AES key register 5 (AES_KEYR5)	511
20.7.15	AES key register 6 (AES_KEYR6)	511
20.7.16	AES key register 7 (AES_KEYR7)	512
20.7.17	AES suspend registers (AES_SUSPxR)	512
20.7.18	AES register map	513
<b>21</b>	<b>Advanced-control timer (TIM1)</b>	<b>515</b>
21.1	TIM1 introduction	515
21.2	TIM1 main features	516
21.3	TIM1 functional description	518
21.3.1	Time-base unit	518
21.3.2	Counter modes	520
21.3.3	Repetition counter	531
21.3.4	External trigger input	533
21.3.5	Clock selection	534
21.3.6	Capture/compare channels	538
21.3.7	Input capture mode	540
21.3.8	PWM input mode	541

21.3.9	Forced output mode	542
21.3.10	Output compare mode	543
21.3.11	PWM mode	544
21.3.12	Asymmetric PWM mode	547
21.3.13	Combined PWM mode	548
21.3.14	Combined 3-phase PWM mode	549
21.3.15	Complementary outputs and dead-time insertion	550
21.3.16	Using the break function	552
21.3.17	Bidirectional break inputs	558
21.3.18	Clearing the OCxREF signal on an external event	560
21.3.19	6-step PWM generation	561
21.3.20	One-pulse mode	562
21.3.21	Retriggerable one pulse mode	563
21.3.22	Encoder interface mode	564
21.3.23	UIF bit remapping	566
21.3.24	Timer input XOR function	567
21.3.25	Interfacing with Hall sensors	567
21.3.26	Timer synchronization	570
21.3.27	ADC synchronization	574
21.3.28	DMA burst mode	574
21.3.29	Debug mode	575
21.4	TIM1 registers	576
21.4.1	TIM1 control register 1 (TIM1_CR1)	576
21.4.2	TIM1 control register 2 (TIM1_CR2)	577
21.4.3	TIM1 slave mode control register (TIM1_SMCR)	580
21.4.4	TIM1 DMA/interrupt enable register (TIM1_DIER)	582
21.4.5	TIM1 status register (TIM1_SR)	584
21.4.6	TIM1 event generation register (TIM1_EGR)	586
21.4.7	TIM1 capture/compare mode register 1 (TIM1_CCMR1)	587
21.4.8	TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)	588
21.4.9	TIM1 capture/compare mode register 2 (TIM1_CCMR2)	591
21.4.10	TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)	592
21.4.11	TIM1 capture/compare enable register (TIM1_CCER)	594

21.4.12	TIM1 counter (TIM1_CNT) .....	597
21.4.13	TIM1 prescaler (TIM1_PSC) .....	597
21.4.14	TIM1 auto-reload register (TIM1_ARR) .....	597
21.4.15	TIM1 repetition counter register (TIM1_RCR) .....	598
21.4.16	TIM1 capture/compare register 1 (TIM1_CCR1) .....	598
21.4.17	TIM1 capture/compare register 2 (TIM1_CCR2) .....	599
21.4.18	TIM1 capture/compare register 3 (TIM1_CCR3) .....	599
21.4.19	TIM1 capture/compare register 4 (TIM1_CCR4) .....	600
21.4.20	TIM1 break and dead-time register (TIM1_BDTR) .....	600
21.4.21	TIM1 DMA control register (TIM1_DCR) .....	604
21.4.22	TIM1 DMA address for full transfer (TIM1_DMAR) .....	605
21.4.23	TIM1 option register 1 (TIM1_OR1) .....	606
21.4.24	TIM1 capture/compare mode register 3 (TIM1_CCMR3) .....	606
21.4.25	TIM1 capture/compare register 5 (TIM1_CCR5) .....	607
21.4.26	TIM1 capture/compare register 6 (TIM1_CCR6) .....	608
21.4.27	TIM1 alternate function option register 1 (TIM1_AF1) .....	609
21.4.28	TIM1 Alternate function register 2 (TIM1_AF2) .....	611
21.4.29	TIM1 timer input selection register (TIM1_TISEL) .....	612
21.4.30	TIM1 register map .....	614
<b>22</b>	<b>General-purpose timers (TIM2/TIM3/TIM4) .....</b>	<b>617</b>
22.1	TIM2/TIM3/TIM4 introduction .....	617
22.2	TIM2/TIM3/TIM4 main features .....	617
22.3	TIM2/TIM3/TIM4 functional description .....	619
22.3.1	Time-base unit .....	619
22.3.2	Counter modes .....	621
22.3.3	Clock selection .....	631
22.3.4	Capture/Compare channels .....	635
22.3.5	Input capture mode .....	637
22.3.6	PWM input mode .....	638



22.3.7	Forced output mode	639
22.3.8	Output compare mode	639
22.3.9	PWM mode	640
22.3.10	Asymmetric PWM mode	644
22.3.11	Combined PWM mode	644
22.3.12	Clearing the OCxREF signal on an external event	645
22.3.13	One-pulse mode	647
22.3.14	Retriggerable one pulse mode	648
22.3.15	Encoder interface mode	649
22.3.16	UIF bit remapping	651
22.3.17	Timer input XOR function	651
22.3.18	Timers and external trigger synchronization	652
22.3.19	Timer synchronization	655
22.3.20	DMA burst mode	660
22.3.21	Debug mode	661
22.4	TIM2/TIM3/TIM4 registers	662
22.4.1	TIMx control register 1 (TIMx_CR1)(x = 2 to 4)	662
22.4.2	TIMx control register 2 (TIMx_CR2)(x = 2 to 4)	663
22.4.3	TIMx slave mode control register (TIMx_SMCR)(x = 2 to 4)	665
22.4.4	TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 4)	668
22.4.5	TIMx status register (TIMx_SR)(x = 2 to 4)	669
22.4.6	TIMx event generation register (TIMx_EGR)(x = 2 to 4)	671
22.4.7	TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 4)	672
22.4.8	TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 4)	674
22.4.9	TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 4)	676
22.4.10	TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 4)	677
22.4.11	TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 4)	678
22.4.12	TIMx counter (TIMx_CNT)(x = 2 to 4)	679
22.4.13	TIMx counter [alternate] (TIMx_CNT)(x = 2 to 4)	680
22.4.14	TIMx prescaler (TIMx_PSC)(x = 2 to 4)	680
22.4.15	TIMx auto-reload register (TIMx_ARR)(x = 2 to 4)	681
22.4.16	TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 4)	681
22.4.17	TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 4)	682
22.4.18	TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 4)	682
22.4.19	TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 4)	683

22.4.20	TIMx DMA control register (TIMx_DCR)(x = 2 to 4)	684
22.4.21	TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 4)	684
22.4.22	TIM2 option register 1 (TIM2_OR1)	684
22.4.23	TIM3 option register 1 (TIM3_OR1)	685
22.4.24	TIM4 option register 1 (TIM4_OR1)	686
22.4.25	TIM2 alternate function option register 1 (TIM2_AF1)	686
22.4.26	TIM3 alternate function option register 1 (TIM3_AF1)	687
22.4.27	TIM4 alternate function option register 1 (TIM4_AF1)	687
22.4.28	TIM2 timer input selection register (TIM2_TISEL)	688
22.4.29	TIM3 timer input selection register (TIM3_TISEL)	688
22.4.30	TIM4 timer input selection register (TIM4_TISEL)	689
22.4.31	TIMx register map	691
<b>23</b>	<b>Basic timers (TIM6/TIM7)</b>	<b>694</b>
23.1	TIM6/TIM7 introduction	694
23.2	TIM6/TIM7 main features	694
23.3	TIM6/TIM7 functional description	695
23.3.1	Time-base unit	695
23.3.2	Counting mode	697
23.3.3	UIF bit remapping	700
23.3.4	Clock source	700
23.3.5	Debug mode	701
23.4	TIM6/TIM7 registers	701
23.4.1	TIMx control register 1 (TIMx_CR1)(x = 6 to 7)	701
23.4.2	TIMx control register 2 (TIMx_CR2)(x = 6 to 7)	703
23.4.3	TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)	703
23.4.4	TIMx status register (TIMx_SR)(x = 6 to 7)	704
23.4.5	TIMx event generation register (TIMx_EGR)(x = 6 to 7)	704
23.4.6	TIMx counter (TIMx_CNT)(x = 6 to 7)	704
23.4.7	TIMx prescaler (TIMx_PSC)(x = 6 to 7)	705
23.4.8	TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)	705
23.4.9	TIMx register map	706
<b>24</b>	<b>General-purpose timers (TIM14)</b>	<b>707</b>
24.1	TIM14 introduction	707
24.2	TIM14 main features	707
24.2.1	TIM14 main features	707

24.3	TIM14 functional description	709
24.3.1	Time-base unit	709
24.3.2	Counter modes	711
24.3.3	Clock selection	714
24.3.4	Capture/compare channels	715
24.3.5	Input capture mode	716
24.3.6	Forced output mode	717
24.3.7	Output compare mode	718
24.3.8	PWM mode	719
24.3.9	One-pulse mode	720
24.3.10	UIF bit remapping	720
24.3.11	Using timer output as trigger for other timers (TIM14)	721
24.3.12	Debug mode	721
24.4	TIM14 registers	722
24.4.1	TIM14 control register 1 (TIM14_CR1)	722
24.4.2	TIM14 Interrupt enable register (TIM14_DIER)	723
24.4.3	TIM14 status register (TIM14_SR)	723
24.4.4	TIM14 event generation register (TIM14_EGR)	724
24.4.5	TIM14 capture/compare mode register 1 (TIM14_CCMR1)	725
24.4.6	TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1)	726
24.4.7	TIM14 capture/compare enable register (TIM14_CCER)	728
24.4.8	TIM14 counter (TIM14_CNT)	729
24.4.9	TIM14 prescaler (TIM14_PSC)	730
24.4.10	TIM14 auto-reload register (TIM14_ARR)	730
24.4.11	TIM14 capture/compare register 1 (TIM14_CCR1)	730
24.4.12	TIM14 timer input selection register (TIM14_TISEL)	731
24.4.13	TIM14 register map	731
<b>25</b>	<b>General-purpose timers (TIM15/TIM16/TIM17)</b>	<b>733</b>
25.1	TIM15/TIM16/TIM17 introduction	733
25.2	TIM15 main features	733
25.3	TIM16/TIM17 main features	734
25.4	TIM15/TIM16/TIM17 functional description	737
25.4.1	Time-base unit	737
25.4.2	Counter modes	739
25.4.3	Repetition counter	743

25.4.4	Clock selection	744
25.4.5	Capture/compare channels	746
25.4.6	Input capture mode	748
25.4.7	PWM input mode (only for TIM15)	749
25.4.8	Forced output mode	750
25.4.9	Output compare mode	751
25.4.10	PWM mode	752
25.4.11	Combined PWM mode (TIM15 only)	753
25.4.12	Complementary outputs and dead-time insertion	754
25.4.13	Using the break function	756
25.4.14	Bidirectional break inputs	761
25.4.15	6-step PWM generation	762
25.4.16	One-pulse mode	764
25.4.17	Retriggerable one pulse mode (TIM15 only)	765
25.4.18	UIF bit remapping	766
25.4.19	Timer input XOR function (TIM15 only)	767
25.4.20	External trigger synchronization (TIM15 only)	768
25.4.21	Slave mode – combined reset + trigger mode	770
25.4.22	DMA burst mode	770
25.4.23	Timer synchronization (TIM15)	772
25.4.24	Using timer output as trigger for other timers (TIM16/TIM17)	772
25.4.25	Debug mode	772
25.5	TIM15 registers	773
25.5.1	TIM15 control register 1 (TIM15_CR1)	773
25.5.2	TIM15 control register 2 (TIM15_CR2)	774
25.5.3	TIM15 slave mode control register (TIM15_SMCR)	776
25.5.4	TIM15 DMA/interrupt enable register (TIM15_DIER)	777
25.5.5	TIM15 status register (TIM15_SR)	778
25.5.6	TIM15 event generation register (TIM15_EGR)	780
25.5.7	TIM15 capture/compare mode register 1 (TIM15_CCMR1)	781
25.5.8	TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)	782
25.5.9	TIM15 capture/compare enable register (TIM15_CCER)	785
25.5.10	TIM15 counter (TIM15_CNT)	788
25.5.11	TIM15 prescaler (TIM15_PSC)	788
25.5.12	TIM15 auto-reload register (TIM15_ARR)	788
25.5.13	TIM15 repetition counter register (TIM15_RCR)	789

25.5.14	TIM15 capture/compare register 1 (TIM15_CCR1)	789
25.5.15	TIM15 capture/compare register 2 (TIM15_CCR2)	790
25.5.16	TIM15 break and dead-time register (TIM15_BDTR)	790
25.5.17	TIM15 DMA control register (TIM15_DCR)	793
25.5.18	TIM15 DMA address for full transfer (TIM15_DMAR)	793
25.5.19	TIM15 alternate register 1 (TIM15_AF1)	794
25.5.20	TIM15 input selection register (TIM15_TISEL)	795
25.5.21	TIM15 register map	796
<b>25.6</b>	<b>TIM16/TIM17 registers</b>	<b>799</b>
25.6.1	TIMx control register 1 (TIMx_CR1)(x = 16 to 17)	799
25.6.2	TIMx control register 2 (TIMx_CR2)(x = 16 to 17)	800
25.6.3	TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)	801
25.6.4	TIMx status register (TIMx_SR)(x = 16 to 17)	802
25.6.5	TIMx event generation register (TIMx_EGR)(x = 16 to 17)	803
25.6.6	TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16 to 17)	804
25.6.7	TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)	805
25.6.8	TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)	807
25.6.9	TIMx counter (TIMx_CNT)(x = 16 to 17)	809
25.6.10	TIMx prescaler (TIMx_PSC)(x = 16 to 17)	810
25.6.11	TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)	810
25.6.12	TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)	811
25.6.13	TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)	811
25.6.14	TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)	812
25.6.15	TIMx DMA control register (TIMx_DCR)(x = 16 to 17)	815
25.6.16	TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)	815
25.6.17	TIM16 alternate function register 1 (TIM16_AF1)	816
25.6.18	TIM16 input selection register (TIM16_TISEL)	817
25.6.19	TIM17 alternate function register 1 (TIM17_AF1)	818
25.6.20	TIM17 input selection register (TIM17_TISEL)	819
25.6.21	TIM16/TIM17 register map	821
<b>26</b>	<b>Low-power timer (LPTIM)</b>	<b>823</b>
26.1	Introduction	823
26.2	LPTIM main features	823
26.3	LPTIM implementation	824

26.4	LPTIM functional description	824
26.4.1	LPTIM block diagram	824
26.4.2	LPTIM pins and internal signals	825
26.4.3	LPTIM input and trigger mapping	825
26.4.4	LPTIM reset and clocks	827
26.4.5	Glitch filter	827
26.4.6	Prescaler	828
26.4.7	Trigger multiplexer	828
26.4.8	Operating mode	829
26.4.9	Timeout function	831
26.4.10	Waveform generation	831
26.4.11	Register update	832
26.4.12	Counter mode	833
26.4.13	Timer enable	833
26.4.14	Timer counter reset	834
26.4.15	Encoder mode	834
26.4.16	Debug mode	836
26.5	LPTIM low-power modes	836
26.6	LPTIM interrupts	837
26.7	LPTIM registers	837
26.7.1	LPTIM interrupt and status register (LPTIM_ISR)	838
26.7.2	LPTIM interrupt clear register (LPTIM_ICR)	839
26.7.3	LPTIM interrupt enable register (LPTIM_IER)	839
26.7.4	LPTIM configuration register (LPTIM_CFGR)	840
26.7.5	LPTIM control register (LPTIM_CR)	843
26.7.6	LPTIM compare register (LPTIM_CMP)	845
26.7.7	LPTIM autoreload register (LPTIM_ARR)	845
26.7.8	LPTIM counter register (LPTIM_CNT)	846
26.7.9	LPTIM configuration register 2 (LPTIM_CFGR2)	846
26.7.10	LPTIM register map	848
<b>27</b>	<b>Infrared interface (IRTIM)</b>	<b>849</b>
<b>28</b>	<b>Independent watchdog (IWDG)</b>	<b>850</b>
28.1	Introduction	850
28.2	IWDG main features	850

28.3	IWDG functional description	850
28.3.1	IWDG block diagram	850
28.3.2	Window option	851
28.3.3	Hardware watchdog	852
28.3.4	Register access protection	852
28.3.5	Debug mode	852
28.4	IWDG registers	853
28.4.1	IWDG key register (IWDG_KR)	853
28.4.2	IWDG prescaler register (IWDG_PR)	854
28.4.3	IWDG reload register (IWDG_RLR)	855
28.4.4	IWDG status register (IWDG_SR)	856
28.4.5	IWDG window register (IWDG_WINR)	857
28.4.6	IWDG register map	858
<b>29</b>	<b>System window watchdog (WWDG)</b>	<b>859</b>
29.1	Introduction	859
29.2	WWDG main features	859
29.3	WWDG functional description	859
29.3.1	WWDG block diagram	860
29.3.2	Enabling the watchdog	860
29.3.3	Controlling the down-counter	860
29.3.4	How to program the watchdog timeout	860
29.3.5	Debug mode	862
29.4	WWDG interrupts	862
29.5	WWDG registers	862
29.5.1	WWDG control register (WWDG_CR)	862
29.5.2	WWDG configuration register (WWDG_CFR)	863
29.5.3	WWDG status register (WWDG_SR)	864
29.5.4	WWDG register map	864
<b>30</b>	<b>Real-time clock (RTC)</b>	<b>865</b>
30.1	Introduction	865
30.2	RTC main features	865
30.3	RTC functional description	866
30.3.1	RTC block diagram	866
30.3.2	RTC pins and internal signals	867

30.3.3	GPIOs controlled by the RTC and TAMP	868
30.3.4	Clock and prescalers	870
30.3.5	Real-time clock and calendar	871
30.3.6	Programmable alarms	872
30.3.7	Periodic auto-wake-up	872
30.3.8	RTC initialization and configuration	873
30.3.9	Reading the calendar	875
30.3.10	Resetting the RTC	876
30.3.11	RTC synchronization	876
30.3.12	RTC reference clock detection	877
30.3.13	RTC smooth digital calibration	877
30.3.14	Timestamp function	879
30.3.15	Calibration clock output	880
30.3.16	Tamper and alarm output	880
30.4	RTC low-power modes	881
30.5	RTC interrupts	882
30.6	RTC registers	882
30.6.1	RTC time register (RTC_TR)	882
30.6.2	RTC date register (RTC_DR)	883
30.6.3	RTC sub second register (RTC_SSR)	884
30.6.4	RTC initialization control and status register (RTC_ICSR)	884
30.6.5	RTC prescaler register (RTC_PRER)	886
30.6.6	RTC wake-up timer register (RTC_WUTR)	887
30.6.7	RTC control register (RTC_CR)	887
30.6.8	RTC write protection register (RTC_WPR)	890
30.6.9	RTC calibration register (RTC_CALR)	891
30.6.10	RTC shift control register (RTC_SHIFTR)	892
30.6.11	RTC timestamp time register (RTC_TSTR)	893
30.6.12	RTC timestamp date register (RTC_TSDR)	893
30.6.13	RTC timestamp sub second register (RTC_TSSSR)	894
30.6.14	RTC alarm A register (RTC_ALRMAR)	895
30.6.15	RTC alarm A sub second register (RTC_ALRMASR)	896
30.6.16	RTC alarm B register (RTC_ALRMBR)	897
30.6.17	RTC alarm B sub second register (RTC_ALRMBSSR)	898
30.6.18	RTC status register (RTC_SR)	898
30.6.19	RTC masked interrupt status register (RTC_MISR)	899
30.6.20	RTC status clear register (RTC_SCR)	900



	30.6.21	RTC register map	902
<b>31</b>		<b>Tamper and backup registers (TAMP)</b>	<b>904</b>
	31.1	Introduction	904
	31.2	TAMP main features	904
	31.3	TAMP functional description	905
	31.3.1	TAMP block diagram	905
	31.3.2	TAMP pins and internal signals	906
	31.3.3	TAMP register write protection	906
	31.3.4	Tamper detection	907
	31.4	TAMP low-power modes	909
	31.5	TAMP interrupts	909
	31.6	TAMP registers	909
	31.6.1	TAMP control register 1 (TAMP_CR1)	910
	31.6.2	TAMP control register 2 (TAMP_CR2)	911
	31.6.3	TAMP filter control register (TAMP_FLTCR)	912
	31.6.4	TAMP interrupt enable register (TAMP_IER)	913
	31.6.5	TAMP status register (TAMP_SR)	914
	31.6.6	TAMP masked interrupt status register (TAMP_MISR)	915
	31.6.7	TAMP status clear register (TAMP_SCR)	916
	31.6.8	TAMP backup x register (TAMP_BKPxR)	917
	31.6.9	TAMP register map	918
<b>32</b>		<b>Inter-integrated circuit interface (I2C)</b>	<b>919</b>
	32.1	Introduction	919
	32.2	I2C main features	919
	32.3	I2C implementation	920
	32.4	I2C functional description	920
	32.4.1	I2C block diagram	921
	32.4.2	I2C pins and internal signals	922
	32.4.3	I2C clock requirements	922
	32.4.4	I2C mode selection	922
	32.4.5	I2C initialization	923
	32.4.6	I2C reset	927
	32.4.7	I2C data transfer	928
	32.4.8	I2C target mode	930

32.4.9	I2C controller mode	939
32.4.10	I2C_TIMINGR register configuration examples	950
32.4.11	SMBus specific features	952
32.4.12	SMBus initialization	955
32.4.13	SMBus I2C_TIMEOUTR register configuration examples	957
32.4.14	SMBus target mode	958
32.4.15	SMBus controller mode	961
32.4.16	Wake-up from Stop mode on address match	964
32.4.17	Error conditions	965
32.5	I2C in low-power modes	967
32.6	I2C interrupts	967
32.7	I2C DMA requests	968
32.7.1	Transmission using DMA	968
32.7.2	Reception using DMA	968
32.8	I2C debug modes	969
32.9	I2C registers	969
32.9.1	I2C control register 1 (I2C_CR1)	969
32.9.2	I2C control register 2 (I2C_CR2)	972
32.9.3	I2C own address 1 register (I2C_OAR1)	974
32.9.4	I2C own address 2 register (I2C_OAR2)	975
32.9.5	I2C timing register (I2C_TIMINGR)	976
32.9.6	I2C timeout register (I2C_TIMEOUTR)	977
32.9.7	I2C interrupt and status register (I2C_ISR)	978
32.9.8	I2C interrupt clear register (I2C_ICR)	980
32.9.9	I2C PEC register (I2C_PECR)	981
32.9.10	I2C receive data register (I2C_RXDR)	981
32.9.11	I2C transmit data register (I2C_TXDR)	982
32.9.12	I2C register map	983
<b>33</b>	<b>Universal synchronous receiver transmitter (USART)</b>	<b>984</b>
33.1	USART introduction	984
33.2	USART main features	985
33.3	USART extended features	986
33.4	USART implementation	986
33.5	USART functional description	988
33.5.1	USART block diagram	988

33.5.2	USART signals	989
33.5.3	USART character description	990
33.5.4	USART FIFOs and thresholds	992
33.5.5	USART transmitter	992
33.5.6	USART receiver	996
33.5.7	USART baud rate generation	1003
33.5.8	Tolerance of the USART receiver to clock deviation	1004
33.5.9	USART auto baud rate detection	1006
33.5.10	USART multiprocessor communication	1008
33.5.11	USART Modbus communication	1010
33.5.12	USART parity control	1011
33.5.13	USART LIN (local interconnection network) mode	1012
33.5.14	USART synchronous mode	1014
33.5.15	USART single-wire half-duplex communication	1018
33.5.16	USART receiver timeout	1018
33.5.17	USART smartcard mode	1019
33.5.18	USART IrDA SIR ENDEC block	1023
33.5.19	Continuous communication using USART and DMA	1026
33.5.20	RS232 hardware flow control and RS485 Driver Enable	1028
33.5.21	USART low-power management	1031
33.6	USART in low-power modes	1034
33.7	USART interrupts	1035
33.8	USART registers	1036
33.8.1	USART control register 1 (USART_CR1)	1036
33.8.2	USART control register 1 [alternate] (USART_CR1)	1039
33.8.3	USART control register 2 (USART_CR2)	1043
33.8.4	USART control register 3 (USART_CR3)	1047
33.8.5	USART baud rate register (USART_BRR)	1051
33.8.6	USART guard time and prescaler register (USART_GTPR)	1051
33.8.7	USART receiver timeout register (USART_RTOR)	1052
33.8.8	USART request register (USART_RQR)	1053
33.8.9	USART interrupt and status register (USART_ISR)	1054
33.8.10	USART interrupt and status register [alternate] (USART_ISR)	1060
33.8.11	USART interrupt flag clear register (USART_ICR)	1065
33.8.12	USART receive data register (USART_RDR)	1067
33.8.13	USART transmit data register (USART_TDR)	1067
33.8.14	USART prescaler register (USART_PRESC)	1068

	33.8.15	USART register map	1069
<b>34</b>		<b>Low-power universal asynchronous receiver transmitter (LPUART)</b>	<b>1071</b>
	34.1	LPUART introduction	1071
	34.2	LPUART main features	1072
	34.3	LPUART implementation	1073
	34.4	LPUART functional description	1074
	34.4.1	LPUART block diagram	1074
	34.4.2	LPUART signals	1075
	34.4.3	LPUART character description	1076
	34.4.4	LPUART FIFOs and thresholds	1077
	34.4.5	LPUART transmitter	1078
	34.4.6	LPUART receiver	1081
	34.4.7	LPUART baud rate generation	1085
	34.4.8	Tolerance of the LPUART receiver to clock deviation	1086
	34.4.9	LPUART multiprocessor communication	1087
	34.4.10	LPUART parity control	1089
	34.4.11	LPUART single-wire half-duplex communication	1090
	34.4.12	Continuous communication using DMA and LPUART	1090
	34.4.13	RS232 hardware flow control and RS485 Driver Enable	1093
	34.4.14	LPUART low-power management	1095
	34.5	LPUART in low-power modes	1098
	34.6	LPUART interrupts	1099
	34.7	LPUART registers	1100
	34.7.1	LPUART control register 1 (LPUART_CR1)	1100
	34.7.2	LPUART control register 1 [alternate] (LPUART_CR1)	1103
	34.7.3	LPUART control register 2 (LPUART_CR2)	1106
	34.7.4	LPUART control register 3 (LPUART_CR3)	1108
	34.7.5	LPUART baud rate register (LPUART_BRR)	1111
	34.7.6	LPUART request register (LPUART_RQR)	1111
	34.7.7	LPUART interrupt and status register (LPUART_ISR)	1112
	34.7.8	LPUART interrupt and status register [alternate] (LPUART_ISR)	1116
	34.7.9	LPUART interrupt flag clear register (LPUART_ICR)	1119
	34.7.10	LPUART receive data register (LPUART_RDR)	1120
	34.7.11	LPUART transmit data register (LPUART_TDR)	1120

	34.7.12	LPUART prescaler register (LPUART_PRESC) .....	1121
	34.7.13	LPUART register map .....	1122
<b>35</b>		<b>Serial peripheral interface / integrated interchip sound (SPI/I2S) .</b>	<b>1124</b>
	35.1	Introduction .....	1124
	35.2	SPI main features .....	1124
	35.3	I2S main features .....	1125
	35.4	SPI/I2S implementation .....	1125
	35.5	SPI functional description .....	1126
	35.5.1	General description .....	1126
	35.5.2	Communications between one master and one slave .....	1127
	35.5.3	Standard multislave communication .....	1129
	35.5.4	Multimaster communication .....	1130
	35.5.5	Slave select (NSS) pin management .....	1131
	35.5.6	Communication formats .....	1132
	35.5.7	Configuration of SPI .....	1134
	35.5.8	Procedure for enabling SPI .....	1135
	35.5.9	Data transmission and reception procedures .....	1135
	35.5.10	SPI status flags .....	1145
	35.5.11	SPI error flags .....	1146
	35.5.12	NSS pulse mode .....	1147
	35.5.13	TI mode .....	1147
	35.5.14	CRC calculation .....	1148
	35.6	SPI interrupts .....	1150
	35.7	I2S functional description .....	1151
	35.7.1	I2S general description .....	1151
	35.7.2	Supported audio protocols .....	1152
	35.7.3	Start-up description .....	1159
	35.7.4	Clock generator .....	1161
	35.7.5	I <sup>2</sup> S master mode .....	1164
	35.7.6	I <sup>2</sup> S slave mode .....	1165
	35.7.7	I2S status flags .....	1167
	35.7.8	I2S error flags .....	1168
	35.7.9	DMA features .....	1169
	35.8	I2S interrupts .....	1169
	35.9	SPI and I2S registers .....	1170

35.9.1	SPI control register 1 (SPIx_CR1) .....	1170
35.9.2	SPI control register 2 (SPIx_CR2) .....	1172
35.9.3	SPI status register (SPIx_SR) .....	1174
35.9.4	SPI data register (SPIx_DR) .....	1176
35.9.5	SPI CRC polynomial register (SPIx_CRCPR) .....	1176
35.9.6	SPI Rx CRC register (SPIx_RXCRCR) .....	1176
35.9.7	SPI Tx CRC register (SPIx_TXCRCR) .....	1177
35.9.8	SPIx_I2S configuration register (SPIx_I2SCFGR) .....	1177
35.9.9	SPIx_I2S prescaler register (SPIx_I2SPR) .....	1179
35.9.10	SPI/I2S register map .....	1181
<b>36</b>	<b>FD controller area network (FDCAN) .....</b>	<b>1182</b>
36.1	Introduction .....	1182
36.2	FDCAN main features .....	1184
36.3	FDCAN functional description .....	1185
36.3.1	FDCAN block diagram .....	1185
36.3.2	FDCAN pins and internal signals .....	1186
36.3.3	Bit timing .....	1187
36.3.4	Operating modes .....	1188
36.3.5	Error management .....	1197
36.3.6	Message RAM .....	1198
36.3.7	FIFO acknowledge handling .....	1207
36.3.8	FDCAN Rx FIFO element .....	1207
36.3.9	FDCAN Tx buffer element .....	1209
36.3.10	FDCAN Tx event FIFO element .....	1211
36.3.11	FDCAN standard message ID filter element .....	1212
36.3.12	FDCAN extended message ID filter element .....	1213
36.4	FDCAN registers .....	1215
36.4.1	FDCAN core release register (FDCAN_CREL) .....	1215
36.4.2	FDCAN endian register (FDCAN_ENDN) .....	1215
36.4.3	FDCAN data bit timing and prescaler register (FDCAN_DBTP) .....	1215
36.4.4	FDCAN test register (FDCAN_TEST) .....	1216
36.4.5	FDCAN RAM watchdog register (FDCAN_RWD) .....	1217
36.4.6	FDCAN CC control register (FDCAN_CCCR) .....	1218
36.4.7	FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) .....	1219
36.4.8	FDCAN timestamp counter configuration register (FDCAN_TSCC) .....	1221
36.4.9	FDCAN timestamp counter value register (FDCAN_TSCV) .....	1221

36.4.10	FDCAN timeout counter configuration register (FDCAN_TOCC) . . .	1222
36.4.11	FDCAN timeout counter value register (FDCAN_TOCV) . . . . .	1223
36.4.12	FDCAN error counter register (FDCAN_ECR) . . . . .	1223
36.4.13	FDCAN protocol status register (FDCAN_PSR) . . . . .	1224
36.4.14	FDCAN transmitter delay compensation register (FDCAN_TDCR) . .	1226
36.4.15	FDCAN interrupt register (FDCAN_IR) . . . . .	1226
36.4.16	FDCAN interrupt enable register (FDCAN_IE) . . . . .	1229
36.4.17	FDCAN interrupt line select register (FDCAN_ILS) . . . . .	1231
36.4.18	FDCAN interrupt line enable register (FDCAN_ILE) . . . . .	1232
36.4.19	FDCAN global filter configuration register (FDCAN_RXGFC) . . . . .	1232
36.4.20	FDCAN extended ID and mask register (FDCAN_XIDAM) . . . . .	1234
36.4.21	FDCAN high-priority message status register (FDCAN_HPMS) . . . .	1234
36.4.22	FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) . . . . .	1235
36.4.23	CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) . . . . .	1236
36.4.24	FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) . . . . .	1236
36.4.25	FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) . . . . .	1237
36.4.26	FDCAN Tx buffer configuration register (FDCAN_TXBC) . . . . .	1237
36.4.27	FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) . . . . .	1238
36.4.28	FDCAN Tx buffer request pending register (FDCAN_TXBRP) . . . . .	1238
36.4.29	FDCAN Tx buffer add request register (FDCAN_TXBAR) . . . . .	1239
36.4.30	FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . .	1240
36.4.31	FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) .	1240
36.4.32	FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . .	1241
36.4.33	FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) . . . . .	1241
36.4.34	FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) . . . . .	1242
36.4.35	FDCAN Tx event FIFO status register (FDCAN_TXEFS) . . . . .	1242
36.4.36	FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . .	1243
36.4.37	FDCAN CFG clock divider register (FDCAN_CKDIV) . . . . .	1243
36.4.38	FDCAN register map . . . . .	1244
<b>37</b>	<b>Universal serial bus full-speed host/device interface (USB) . . . . .</b>	<b>1248</b>
37.1	Introduction . . . . .	1248
37.2	USB main features . . . . .	1248
37.3	USB implementation . . . . .	1248
37.4	USB functional description . . . . .	1249

37.4.1	USB block diagram	1249
37.4.2	USB pins and internal signals	1249
37.4.3	USB reset and clocks	1250
37.4.4	General description and Device mode functionality	1250
37.4.5	Description of USB blocks used in both Device and Host modes	1251
37.4.6	Description of host frame scheduler (HFS) specific to Host mode	1252
37.5	Programming considerations for Device and Host modes	1253
37.5.1	Generic USB Device programming	1253
37.5.2	System and power-on reset	1254
37.5.3	Double-buffered endpoints and usage in Device mode	1261
37.5.4	Double buffered channels: usage in Host mode	1263
37.5.5	Isochronous transfers in Device mode	1264
37.5.6	Isochronous transfers in Host mode	1265
37.5.7	Suspend/resume events	1266
37.6	USB registers	1270
37.6.1	USB control register (USB_CNTR)	1270
37.6.2	USB interrupt status register (USB_ISTR)	1273
37.6.3	USB frame number register (USB_FNR)	1276
37.6.4	USB Device address (USB_DADDR)	1277
37.6.5	USB LPM control and status register (USB_LPMCSR)	1278
37.6.6	USB battery charging detector (USB_BCDR)	1278
37.6.7	USB endpoint/channel n register (USB_CHEPnR)	1280
37.6.8	USB register map	1289
37.7	USBSRAM registers	1290
37.7.1	Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n)	1291
37.7.2	Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n)	1291
37.7.3	Channel/endpoint receive buffer descriptor n (USB_CHEP_RXTXBD_n)	1293
37.7.4	Channel/endpoint transmit buffer descriptor n [alternate] (USB_CHEP_RXTXBD_n)	1294
37.7.5	USBSRAM register map	1295
<b>38</b>	<b>USB Type-C<sup>®</sup>/USB Power Delivery interface (UCPD)</b>	<b>1296</b>
38.1	Introduction	1296
38.2	UCPD main features	1296
38.3	UCPD implementation	1297



38.4	UCPD functional description	1297
38.4.1	UCPD block diagram	1298
38.4.2	UCPD reset and clocks	1299
38.4.3	Physical layer protocol	1300
38.4.4	UCPD BMC transmitter	1306
38.4.5	UCPD BMC receiver	1308
38.4.6	UCPD Type-C pull-ups (Rp) and pull-downs (Rd)	1309
38.4.7	UCPD Type-C voltage monitoring and de-bouncing	1310
38.4.8	UCPD fast role swap (FRS)	1310
38.4.9	UCPD DMA Interface	1310
38.4.10	Wake-up from Stop mode	1310
38.5	UCPD programming sequences	1311
38.5.1	Initialization phase	1311
38.5.2	Type-C state machine handling	1311
38.5.3	USB PD transmit	1313
38.5.4	USB PD receive	1314
38.6	UCPD low-power modes	1315
38.7	UCPD interrupts	1316
38.8	UCPD registers	1317
38.8.1	UCPD configuration register 1 (UCPD_CFGR1)	1317
38.8.2	UCPD configuration register 2 (UCPD_CFGR2)	1319
38.8.3	UCPD control register (UCPD_CR)	1320
38.8.4	UCPD interrupt mask register (UCPD_IMR)	1322
38.8.5	UCPD status register (UCPD_SR)	1324
38.8.6	UCPD interrupt clear register (UCPD_ICR)	1327
38.8.7	UCPD Tx ordered set type register (UCPD_TX_ORDSETR)	1328
38.8.8	UCPD Tx payload size register (UCPD_TX_PAYSZR)	1328
38.8.9	UCPD Tx data register (UCPD_TXDR)	1329
38.8.10	UCPD Rx ordered set register (UCPD_RX_ORDSETR)	1329
38.8.11	UCPD Rx payload size register (UCPD_RX_PAYSZR)	1330
38.8.12	UCPD receive data register (UCPD_RXDR)	1331
38.8.13	UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1)	1331
38.8.14	UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2)	1332
38.8.15	UCPD register map	1332

<b>39</b>	<b>HDMI-CEC controller (CEC)</b> .....	<b>1335</b>
39.1	HDMI-CEC introduction .....	1335
39.2	HDMI-CEC controller main features .....	1335
39.3	HDMI-CEC functional description .....	1336
39.3.1	HDMI-CEC pin .....	1336
39.3.2	HDMI-CEC block diagram .....	1336
39.3.3	Message description .....	1336
39.3.4	Bit timing .....	1337
39.4	Arbitration .....	1338
39.4.1	SFT option bit .....	1339
39.5	Error handling .....	1340
39.5.1	Bit error .....	1340
39.5.2	Message error .....	1340
39.5.3	Bit rising error (BRE) .....	1340
39.5.4	Short bit period error (SBPE) .....	1341
39.5.5	Long bit period error (LBPE) .....	1341
39.5.6	Transmission error detection (TXERR) .....	1342
39.6	HDMI-CEC interrupts .....	1344
39.7	HDMI-CEC registers .....	1345
39.7.1	CEC control register (CEC_CR) .....	1345
39.7.2	CEC configuration register (CEC_CFGR) .....	1346
39.7.3	CEC Tx data register (CEC_TXDR) .....	1348
39.7.4	CEC Rx data register (CEC_RXDR) .....	1348
39.7.5	CEC interrupt and status register (CEC_ISR) .....	1348
39.7.6	CEC interrupt enable register (CEC_IER) .....	1350
39.7.7	HDMI-CEC register map .....	1352
<b>40</b>	<b>Debug support (DBG)</b> .....	<b>1353</b>
40.1	Overview .....	1353
40.2	Reference Arm documentation .....	1354
40.3	Pinout and debug port pins .....	1354
40.3.1	SWD port pins .....	1354
40.3.2	SW-DP pin assignment .....	1354
40.3.3	Internal pull-up & pull-down on SWD pins .....	1355
40.4	ID codes and locking mechanism .....	1355
40.5	SWD port .....	1355

40.5.1	SWD protocol introduction	1355
40.5.2	SWD protocol sequence	1355
40.5.3	SW-DP state machine (reset, idle states, ID code)	1356
40.5.4	DP and AP read/write accesses	1357
40.5.5	SW-DP registers	1357
40.5.6	SW-AP registers	1358
40.6	Core debug	1359
40.7	BPU (Break Point Unit)	1359
40.7.1	BPU functionality	1360
40.8	DWT (Data Watchpoint)	1360
40.8.1	DWT functionality	1360
40.8.2	DWT Program Counter Sample Register	1360
40.9	MCU debug component (DBG)	1360
40.9.1	Debug support for low-power modes	1360
40.9.2	Debug support for timers, watchdog and I <sup>2</sup> C	1361
40.10	DBG registers	1361
40.10.1	DBG device ID code register (DBG_IDCODE)	1361
40.10.2	DBG configuration register (DBG_CR)	1362
40.10.3	DBG APB freeze register 1 (DBG_APB_FZ1)	1363
40.10.4	DBG APB freeze register 2 (DBG_APB_FZ2)	1365
40.10.5	DBG register map	1366
<b>41</b>	<b>Device electronic signature</b>	<b>1368</b>
41.1	Unique device ID register (96 bits)	1368
41.2	Flash memory size data register	1369
41.3	Package data register	1369
<b>42</b>	<b>Important security notice</b>	<b>1371</b>
<b>43</b>	<b>Revision history</b>	<b>1372</b>

## List of tables

Table 1.	Peripherals versus devices . . . . .	56
Table 2.	STM32G0B1xx and STM32G0C1xx memory boundary addresses . . . . .	62
Table 3.	STM32G071xx and STM32G081xx memory boundary addresses . . . . .	62
Table 4.	STM32G051xx and STM32G061xx memory boundary addresses . . . . .	62
Table 5.	STM32G031xx and STM32G041xx memory boundary addresses . . . . .	63
Table 6.	STM32G0x1 peripheral register boundary addresses . . . . .	64
Table 7.	SRAM size . . . . .	66
Table 8.	Boot modes . . . . .	68
Table 9.	Flash memory organization for single-bank devices . . . . .	71
Table 10.	Flash memory organization for 256 KB dual-bank devices . . . . .	72
Table 11.	Flash memory organization for 512 KB devices . . . . .	73
Table 12.	Flash memory bank mapping . . . . .	73
Table 13.	Number of wait states according to flash memory clock (HCLK) frequency . . . . .	75
Table 14.	Page erase overview . . . . .	78
Table 15.	Mass erase overview . . . . .	78
Table 16.	Option byte format . . . . .	82
Table 17.	Organization of option bytes . . . . .	83
Table 18.	Flash memory read protection status . . . . .	86
Table 20.	Access status versus protection level and execution modes . . . . .	88
Table 23.	Securable memory erase at RDP Level 1 to Level 0 change . . . . .	92
Table 24.	FLASH interrupt requests . . . . .	93
Table 25.	FLASH register map and reset values . . . . .	112
Table 26.	Low-power mode summary . . . . .	123
Table 27.	Functionalities depending on the working mode . . . . .	124
Table 28.	Low-power run . . . . .	127
Table 29.	Sleep mode summary . . . . .	128
Table 30.	Low-power sleep mode summary . . . . .	129
Table 31.	Stop 0 mode summary . . . . .	131
Table 32.	Stop 1 mode summary . . . . .	132
Table 33.	Standby mode summary . . . . .	134
Table 34.	Shutdown mode summary . . . . .	136
Table 35.	PWR register map and reset values . . . . .	153
Table 36.	Clock source frequency . . . . .	166
Table 37.	RCC register map and reset values . . . . .	213
Table 38.	CRS features . . . . .	217
Table 39.	CRS internal input/output signals . . . . .	218
Table 40.	CRS interconnection . . . . .	219
Table 41.	Effect of low-power modes on CRS . . . . .	222
Table 42.	Interrupt control bits . . . . .	222
Table 43.	CRS register map and reset values . . . . .	227
Table 44.	Port bit configuration table . . . . .	229
Table 45.	Effect of low-power modes on the GPIO . . . . .	238
Table 46.	GPIO register map and reset values . . . . .	245
Table 47.	SYSCFG register map and reset values . . . . .	265
Table 48.	Interconnect matrix . . . . .	268
Table 49.	DMA implementation . . . . .	276
Table 50.	DMA internal input/output signals . . . . .	277
Table 51.	Programmable data width and endian behavior (when PINC = MINC = 1) . . . . .	283

Table 52.	DMA interrupt requests . . . . .	285
Table 53.	DMA register map and reset values . . . . .	293
Table 54.	DMAMUX instantiation . . . . .	297
Table 55.	DMAMUX: assignment of multiplexer inputs to resources . . . . .	298
Table 56.	DMAMUX: assignment of trigger inputs to resources . . . . .	298
Table 57.	DMAMUX: assignment of synchronization inputs to resources . . . . .	299
Table 58.	DMAMUX signals . . . . .	301
Table 59.	DMAMUX interrupts . . . . .	305
Table 60.	DMAMUX register map and reset values . . . . .	310
Table 61.	Vector table . . . . .	312
Table 62.	EXTI signal overview . . . . .	316
Table 63.	EVG pin overview . . . . .	316
Table 64.	EXTI event input configurations and register control . . . . .	318
Table 65.	EXTI line connections . . . . .	320
Table 66.	Masking functionality . . . . .	321
Table 67.	EXTI register map sections . . . . .	322
Table 68.	EXTI controller register map and reset values . . . . .	332
Table 69.	CRC internal input/output signals . . . . .	335
Table 70.	CRC register map and reset values . . . . .	340
Table 71.	ADC input/output pins . . . . .	343
Table 72.	ADC internal input/output signals . . . . .	344
Table 73.	External triggers . . . . .	344
Table 74.	Latency between trigger and start of conversion . . . . .	350
Table 75.	Configuring the trigger polarity . . . . .	357
Table 76.	tSAR timings depending on resolution . . . . .	359
Table 77.	Analog watchdog comparison . . . . .	368
Table 78.	Analog watchdog 1 channel selection . . . . .	368
Table 79.	Maximum output results vs N and M. Grayed values indicates truncation . . . . .	373
Table 80.	ADC interrupts . . . . .	378
Table 81.	ADC register map and reset values . . . . .	397
Table 82.	DAC features . . . . .	401
Table 83.	DAC input/output pins . . . . .	403
Table 84.	DAC internal input/output signals . . . . .	403
Table 85.	DAC interconnection . . . . .	404
Table 86.	Sample and refresh timings . . . . .	411
Table 87.	Channel output modes summary . . . . .	412
Table 88.	Effect of low-power modes on DAC . . . . .	419
Table 89.	DAC interrupts . . . . .	420
Table 90.	DAC register map and reset values . . . . .	435
Table 91.	VREF buffer modes . . . . .	437
Table 92.	VREFBUF register map and reset values . . . . .	439
Table 93.	COMP1 non-inverting input assignment . . . . .	441
Table 94.	COMP1 inverting input assignment . . . . .	442
Table 95.	COMP2 non-inverting input assignment . . . . .	442
Table 96.	COMP2 inverting input assignment . . . . .	442
Table 97.	COMP3 non-inverting input assignment . . . . .	443
Table 98.	COMP3 inverting input assignment . . . . .	443
Table 99.	Comparator behavior in the low power modes . . . . .	446
Table 100.	Interrupt control bits . . . . .	446
Table 101.	COMP register map and reset values . . . . .	453
Table 102.	RNG internal input/output signals . . . . .	455
Table 103.	RNG interrupt requests . . . . .	461

Table 104.	RNG configurations	462
Table 105.	RNG register map and reset map	465
Table 106.	AES internal input/output signals	467
Table 107.	CTR mode initialization vector definition	484
Table 108.	GCM last block definition	486
Table 109.	Initialization of AES_IVRx registers in GCM mode	487
Table 110.	Initialization of AES_IVRx registers in CCM mode	494
Table 111.	Key endianness in AES_KEYRx registers (128- or 256-bit key length)	499
Table 112.	AES interrupt requests	502
Table 113.	Processing latency for ECB, CBC and CTR	502
Table 114.	Processing latency for GCM and CCM (in clock cycles)	503
Table 115.	AES register map and reset values	513
Table 116.	Behavior of timer outputs versus BRK/BRK2 inputs	557
Table 117.	Break protection disarming conditions	559
Table 118.	Counting direction versus encoder signals	565
Table 119.	TIM1 internal trigger connection	582
Table 120.	Output control bits for complementary OCx and OCxN channels with break feature	596
Table 121.	TIM1 register map and reset values	614
Table 122.	Counting direction versus encoder signals	650
Table 123.	TIMx internal trigger connection	668
Table 124.	Output control bit for standard OCx channels	679
Table 125.	TIM2/TIM3/TIM4 register map and reset values	691
Table 126.	TIMx register map and reset values	706
Table 127.	Output control bit for standard OCx channels	729
Table 128.	TIM14 register map and reset values	731
Table 129.	Break protection disarming conditions	761
Table 130.	TIMx Internal trigger connection	777
Table 131.	Output control bits for complementary OCx and OCxN channels with break feature (TIM15)	787
Table 132.	TIM15 register map and reset values	796
Table 133.	Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)	809
Table 134.	TIM16/TIM17 register map and reset values	821
Table 135.	STM32G0x1 LPTIM features	824
Table 136.	LPTIM input/output pins	825
Table 137.	LPTIM internal signals	825
Table 138.	LPTIM1 external trigger connection	825
Table 139.	LPTIM2 external trigger connection	826
Table 140.	LPTIM1 input 1 connection	826
Table 141.	LPTIM1 input 2 connection	826
Table 142.	LPTIM2 input 1 connection	826
Table 143.	Prescaler division ratios	828
Table 144.	Encoder counting scenarios	835
Table 145.	Effect of low-power modes on the LPTIM	836
Table 146.	Interrupt events	837
Table 147.	LPTIM register map and reset values	848
Table 148.	IWDG register map and reset values	858
Table 149.	WWDG register map and reset values	864
Table 150.	RTC input/output pins	867
Table 151.	RTC internal input/output signals	867
Table 152.	RTC interconnection	868
Table 153.	PC13 configuration	868

Table 154.	RTC_OUT mapping	870
Table 155.	Effect of low-power modes on RTC	881
Table 156.	RTC pins functionality over modes	881
Table 157.	Interrupt requests	882
Table 158.	RTC register map and reset values	902
Table 159.	TAMP input/output pins	906
Table 160.	TAMP internal input/output signals	906
Table 161.	TAMP interconnection	906
Table 162.	Effect of low-power modes on TAMP	909
Table 163.	Interrupt requests	909
Table 164.	TAMP register map and reset values	918
Table 165.	I2C implementation	920
Table 166.	I2C input/output pins	922
Table 167.	I2C internal input/output signals	922
Table 168.	Comparison of analog and digital filters	924
Table 169.	I <sup>2</sup> C-bus and SMBus specification data setup and hold times	926
Table 170.	I2C configuration	930
Table 171.	I <sup>2</sup> C-bus and SMBus specification clock timings	941
Table 172.	Timing settings for $f_{I2CCCLK}$ of 8 MHz	951
Table 173.	Timing settings for $f_{I2CCCLK}$ of 16 MHz	951
Table 174.	Timing settings for $f_{I2CCCLK}$ of 48 MHz	952
Table 175.	SMBus timeout specifications	954
Table 176.	SMBus with PEC configuration	956
Table 177.	TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms	957
Table 178.	TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms	957
Table 179.	TIMEOUTA[11:0] for maximum $t_{IDLE}$ of 50 $\mu$ s	957
Table 180.	Effect of low-power modes to I2C	967
Table 181.	I2C interrupt requests	967
Table 182.	I2C register map and reset values	983
Table 183.	Instance implementation on STM32G0x1	986
Table 184.	USART / LPUART features	987
Table 185.	USART input/output pins	990
Table 186.	USART internal input/output signals	990
Table 187.	Noise detection from sampled data	1002
Table 188.	Tolerance of the USART receiver when BRR[3:0] = 0000	1005
Table 189.	Tolerance of the USART receiver when BRR[3:0] is different from 0000	1006
Table 190.	USART frame formats	1011
Table 191.	Effect of low-power modes on the USART	1034
Table 192.	USART interrupt requests	1035
Table 193.	USART register map and reset values	1069
Table 194.	Instance implementation on STM32G0x1	1073
Table 195.	USART / LPUART features	1073
Table 196.	LPUART input/output pins	1075
Table 197.	LPUART internal input/output signals	1075
Table 198.	Error calculation for programmed baud rates at $lpuart\_ker\_ck\_pres = 32.768$ kHz	1085
Table 199.	Error calculation for programmed baud rates at $f_{CK} = 100$ MHz	1086
Table 200.	Tolerance of the LPUART receiver	1087
Table 202.	Effect of low-power modes on the LPUART	1098
Table 203.	LPUART interrupt requests	1099
Table 204.	LPUART register map and reset values	1122
Table 205.	STM32G0x1 SPI and SPI/I2S implementation	1125
Table 206.	SPI interrupt requests	1150



Table 207.	Audio-frequency precision using 48 MHz clock derived from HSE.	1163
Table 208.	I2S interrupt requests	1169
Table 209.	SPI/I2S register map and reset values	1181
Table 210.	CAN subsystem I/O signals	1186
Table 211.	CAN subsystem I/O pins.	1186
Table 212.	DLC coding in FDCAN	1190
Table 213.	Possible configurations for frame transmission	1204
Table 214.	Rx FIFO element	1207
Table 215.	Rx FIFO element description	1207
Table 216.	Tx buffer and FIFO element	1209
Table 217.	Tx buffer element description	1209
Table 218.	Tx event FIFO element.	1211
Table 219.	Tx event FIFO element description.	1211
Table 220.	Standard message ID filter element	1212
Table 221.	Standard message ID filter element field description	1213
Table 222.	Extended message ID filter element.	1213
Table 223.	Extended message ID filter element field description.	1214
Table 224.	FDCAN register map and reset values	1244
Table 225.	STM32G0x1 USB implementation	1248
Table 226.	USB input/output pins.	1249
Table 227.	Double-buffering buffer flag definition.	1262
Table 228.	Bulk double-buffering memory buffers usage (Device mode).	1262
Table 229.	Bulk double-buffering memory buffers usage (Host mode)	1264
Table 230.	Isochronous memory buffers usage	1265
Table 231.	Isochronous memory buffers usage	1266
Table 232.	Resume event detection.	1268
Table 233.	Resume event detection for host	1269
Table 234.	Reception status encoding	1287
Table 235.	Endpoint/channel type encoding.	1287
Table 236.	Endpoint/channel kind meaning	1287
Table 237.	Transmission status encoding	1287
Table 238.	USB register map and reset values	1289
Table 239.	Definition of allocated buffer memory	1292
Table 240.	USBSRAM register map and reset values	1295
Table 241.	UCPD implementation	1297
Table 242.	UCPD signals on pins.	1298
Table 243.	UCPD internal signals.	1299
Table 244.	4b5b symbol encoding table.	1300
Table 245.	Ordered sets	1302
Table 246.	Validation of ordered sets.	1302
Table 247.	Data size.	1303
Table 248.	Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx	1311
Table 249.	Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)	1313
Table 250.	Effect of low power modes on the UCPD	1315
Table 251.	UCPD interrupt requests.	1316
Table 252.	UCPD register map and reset values	1332
Table 253.	HDMI pin.	1336
Table 254.	Error handling timing parameters	1342
Table 255.	TXERR timing parameters	1343
Table 256.	HDMI-CEC interrupts	1344
Table 257.	HDMI-CEC register map and reset values	1352
Table 258.	SW debug port pins	1354



---

Table 259.	Packet request (8-bits) . . . . .	1356
Table 260.	ACK response (3 bits) . . . . .	1356
Table 261.	DATA transfer (33 bits) . . . . .	1356
Table 262.	SW-DP registers . . . . .	1357
Table 263.	32-bit debug port registers addressed through the shifted value A[3:2] . . . . .	1358
Table 264.	Core debug registers . . . . .	1359
Table 265.	DEV_ID and REV_ID field values . . . . .	1361
Table 266.	DBG register map and reset values . . . . .	1366
Table 267.	Document revision history . . . . .	1372

## List of figures

Figure 1.	System architecture	58
Figure 2.	Memory map	61
Figure 3.	Changing read protection (RDP) level	88
Figure 4.	Example of disabling core debug access	92
Figure 5.	Power supply overview	115
Figure 6.	POR, PDR, and BOR thresholds	119
Figure 7.	PVD thresholds	120
Figure 8.	Low-power modes state diagram	122
Figure 9.	Simplified diagram of the reset circuit	156
Figure 10.	Clock tree	161
Figure 11.	HSE/ LSE clock sources	162
Figure 12.	Frequency measurement with TIM14 in capture mode	170
Figure 13.	Frequency measurement with TIM16 in capture mode	171
Figure 14.	Frequency measurement with TIM17 in capture mode	171
Figure 15.	CRS block diagram	218
Figure 16.	CRS counter behavior	220
Figure 17.	Basic structure of an I/O port bit	229
Figure 18.	Input floating/pull up/pull down configurations	234
Figure 19.	Output configuration	235
Figure 20.	Alternate function configuration-	236
Figure 21.	High impedance-analog configuration	236
Figure 22.	DMA block diagram	277
Figure 23.	DMAMUX block diagram	300
Figure 24.	Synchronization mode of the DMAMUX request line multiplexer channel	303
Figure 25.	Event generation of the DMA request line multiplexer channel	303
Figure 26.	EXTI block diagram	316
Figure 27.	Configurable event trigger logic CPU wake-up	318
Figure 28.	Direct event trigger logic CPU wake-up	319
Figure 29.	EXTI GPIO mux	320
Figure 30.	CRC calculation unit block diagram	335
Figure 31.	ADC block diagram	343
Figure 32.	ADC calibration	346
Figure 33.	Calibration factor forcing	347
Figure 34.	Enabling/disabling the ADC	348
Figure 35.	ADC clock scheme	349
Figure 36.	ADC connectivity	351
Figure 37.	Analog-to-digital conversion time	356
Figure 38.	ADC conversion timings	356
Figure 39.	Stopping an ongoing conversion	357
Figure 40.	Single conversions of a sequence, software trigger	360
Figure 41.	Continuous conversion of a sequence, software trigger	360
Figure 42.	Single conversions of a sequence, hardware trigger	361
Figure 43.	Continuous conversions of a sequence, hardware trigger	361
Figure 44.	Data alignment and resolution (oversampling disabled: OVSE = 0)	362
Figure 45.	Example of overrun (OVR)	363
Figure 46.	Wait mode conversion (continuous mode, software trigger)	366
Figure 47.	Behavior with WAIT = 0, AUTOFF = 1	367
Figure 48.	Behavior with WAIT = 1, AUTOFF = 1	367

Figure 49.	Analog watchdog guarded area . . . . .	368
Figure 50.	ADC_AWDx_OUT signal generation . . . . .	370
Figure 51.	ADC_AWDx_OUT signal generation (AWDx flag not cleared by software) . . . . .	370
Figure 52.	ADC_AWDx_OUT signal generation (on a single channel) . . . . .	371
Figure 53.	Analog watchdog threshold update . . . . .	371
Figure 54.	20-bit to 16-bit result truncation . . . . .	372
Figure 55.	Numerical example with 5-bit shift and rounding . . . . .	372
Figure 56.	Triggered oversampling mode (TOVS bit = 1) . . . . .	374
Figure 57.	Temperature sensor and VREFINT channel block diagram . . . . .	375
Figure 58.	VBAT channel block diagram . . . . .	377
Figure 59.	Dual-channel DAC block diagram . . . . .	402
Figure 60.	Data registers in single DAC channel mode . . . . .	405
Figure 61.	Data registers in dual DAC channel mode . . . . .	405
Figure 62.	Timing diagram for conversion with trigger disabled TEN = 0 . . . . .	406
Figure 63.	DAC LFSR register calculation algorithm . . . . .	408
Figure 64.	DAC conversion (SW trigger enabled) with LFSR wave generation . . . . .	408
Figure 65.	DAC triangle wave generation . . . . .	409
Figure 66.	DAC conversion (SW trigger enabled) with triangle wave generation . . . . .	409
Figure 67.	DAC sample and hold mode phase diagram . . . . .	412
Figure 68.	Comparator block diagram . . . . .	441
Figure 69.	Window mode . . . . .	444
Figure 70.	Comparator hysteresis . . . . .	444
Figure 71.	Comparator output blanking . . . . .	445
Figure 72.	RNG block diagram . . . . .	455
Figure 73.	Entropy source model . . . . .	456
Figure 74.	RNG initialization overview . . . . .	458
Figure 75.	AES block diagram . . . . .	467
Figure 76.	ECB encryption and decryption principle . . . . .	469
Figure 77.	CBC encryption and decryption principle . . . . .	470
Figure 78.	CTR encryption and decryption principle . . . . .	471
Figure 79.	GCM encryption and authentication principle . . . . .	472
Figure 80.	GMAC authentication principle . . . . .	472
Figure 81.	CCM encryption and authentication principle . . . . .	473
Figure 82.	Example of suspend mode management . . . . .	477
Figure 83.	ECB encryption . . . . .	478
Figure 84.	ECB decryption . . . . .	478
Figure 85.	CBC encryption . . . . .	479
Figure 86.	CBC decryption . . . . .	479
Figure 87.	ECB/CBC encryption (Mode 1) . . . . .	480
Figure 88.	ECB/CBC decryption (Mode 3) . . . . .	481
Figure 89.	Message construction in CTR mode . . . . .	483
Figure 90.	CTR encryption . . . . .	483
Figure 91.	CTR decryption . . . . .	484
Figure 92.	Message construction in GCM . . . . .	485
Figure 93.	GCM authenticated encryption . . . . .	487
Figure 94.	Message construction in GMAC mode . . . . .	491
Figure 95.	GMAC authentication mode . . . . .	491
Figure 96.	Message construction in CCM mode . . . . .	492
Figure 97.	CCM mode authenticated encryption . . . . .	494
Figure 98.	128-bit block construction with respect to data swap . . . . .	498
Figure 99.	DMA transfer of a 128-bit data block during input phase . . . . .	500
Figure 100.	DMA transfer of a 128-bit data block during output phase . . . . .	501

Figure 101. Advanced-control timer block diagram . . . . .	517
Figure 102. Counter timing diagram with prescaler division change from 1 to 2 . . . . .	519
Figure 103. Counter timing diagram with prescaler division change from 1 to 4 . . . . .	519
Figure 104. Counter timing diagram, internal clock divided by 1 . . . . .	521
Figure 105. Counter timing diagram, internal clock divided by 2 . . . . .	521
Figure 106. Counter timing diagram, internal clock divided by 4 . . . . .	522
Figure 107. Counter timing diagram, internal clock divided by N . . . . .	522
Figure 108. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded) . . . . .	523
Figure 109. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded) . . . . .	523
Figure 110. Counter timing diagram, internal clock divided by 1 . . . . .	525
Figure 111. Counter timing diagram, internal clock divided by 2 . . . . .	525
Figure 112. Counter timing diagram, internal clock divided by 4 . . . . .	526
Figure 113. Counter timing diagram, internal clock divided by N . . . . .	526
Figure 114. Counter timing diagram, update event when repetition counter is not used . . . . .	527
Figure 115. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . . .	528
Figure 116. Counter timing diagram, internal clock divided by 2 . . . . .	529
Figure 117. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . .	529
Figure 118. Counter timing diagram, internal clock divided by N . . . . .	530
Figure 119. Counter timing diagram, update event with ARPE=1 (counter underflow) . . . . .	530
Figure 120. Counter timing diagram, Update event with ARPE=1 (counter overflow) . . . . .	531
Figure 121. Update rate examples depending on mode and TIMx_RCR register settings . . . . .	532
Figure 122. External trigger input block . . . . .	533
Figure 123. TIM1 ETR input circuitry . . . . .	533
Figure 124. Control circuit in normal mode, internal clock divided by 1 . . . . .	534
Figure 125. TI2 external clock connection example . . . . .	535
Figure 126. Control circuit in external clock mode 1 . . . . .	536
Figure 127. External trigger input block . . . . .	536
Figure 128. Control circuit in external clock mode 2 . . . . .	537
Figure 129. Capture/compare channel (example: channel 1 input stage) . . . . .	538
Figure 130. Capture/compare channel 1 main circuit . . . . .	538
Figure 131. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3) . . . . .	539
Figure 132. Output stage of capture/compare channel (channel 4) . . . . .	539
Figure 133. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . . .	540
Figure 134. PWM input mode timing . . . . .	542
Figure 135. Output compare mode, toggle on OC1 . . . . .	544
Figure 136. Edge-aligned PWM waveforms (ARR=8) . . . . .	545
Figure 137. Center-aligned PWM waveforms (ARR=8) . . . . .	546
Figure 138. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . .	548
Figure 139. Combined PWM mode on channel 1 and 3 . . . . .	549
Figure 140. 3-phase combined PWM signals with multiple trigger pulses per period . . . . .	550
Figure 141. Complementary output with dead-time insertion . . . . .	551
Figure 142. Dead-time waveforms with delay greater than the negative pulse . . . . .	551
Figure 143. Dead-time waveforms with delay greater than the positive pulse . . . . .	552
Figure 144. Break and Break2 circuitry overview . . . . .	554
Figure 145. Various output behavior in response to a break event on BRK (OSS1 = 1) . . . . .	556
Figure 146. PWM output state following BRK and BRK2 pins assertion (OSS1=1) . . . . .	557
Figure 147. PWM output state following BRK assertion (OSS1=0) . . . . .	558
Figure 148. Output redirection (BRK2 request not represented) . . . . .	559
Figure 149. Clearing TIMx_OCxREF . . . . .	560
Figure 150. 6-step generation, COM example (OSSR=1) . . . . .	561
Figure 151. Example of one pulse mode . . . . .	562
Figure 152. Retriggerable one pulse mode . . . . .	564

Figure 153. Example of counter operation in encoder interface mode. . . . .	565
Figure 154. Example of encoder interface mode with TI1FP1 polarity inverted. . . . .	566
Figure 155. Measuring time interval between edges on 3 signals . . . . .	567
Figure 156. Example of Hall sensor interface . . . . .	569
Figure 157. Control circuit in reset mode . . . . .	570
Figure 158. Control circuit in Gated mode . . . . .	571
Figure 159. Control circuit in trigger mode . . . . .	572
Figure 160. Control circuit in external clock mode 2 + trigger mode . . . . .	573
Figure 161. General-purpose timer block diagram . . . . .	618
Figure 162. Counter timing diagram with prescaler division change from 1 to 2 . . . . .	620
Figure 163. Counter timing diagram with prescaler division change from 1 to 4 . . . . .	620
Figure 164. Counter timing diagram, internal clock divided by 1 . . . . .	621
Figure 165. Counter timing diagram, internal clock divided by 2 . . . . .	622
Figure 166. Counter timing diagram, internal clock divided by 4 . . . . .	622
Figure 167. Counter timing diagram, internal clock divided by N . . . . .	623
Figure 168. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded). . . . .	623
Figure 169. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded). . . . .	624
Figure 170. Counter timing diagram, internal clock divided by 1 . . . . .	625
Figure 171. Counter timing diagram, internal clock divided by 2 . . . . .	625
Figure 172. Counter timing diagram, internal clock divided by 4 . . . . .	626
Figure 173. Counter timing diagram, internal clock divided by N . . . . .	626
Figure 174. Counter timing diagram, Update event . . . . .	627
Figure 175. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6 . . . . .	628
Figure 176. Counter timing diagram, internal clock divided by 2 . . . . .	629
Figure 177. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36 . . . . .	629
Figure 178. Counter timing diagram, internal clock divided by N . . . . .	630
Figure 179. Counter timing diagram, Update event with ARPE=1 (counter underflow). . . . .	630
Figure 180. Counter timing diagram, Update event with ARPE=1 (counter overflow). . . . .	631
Figure 181. Control circuit in normal mode, internal clock divided by 1 . . . . .	632
Figure 182. TI2 external clock connection example. . . . .	632
Figure 183. Control circuit in external clock mode 1 . . . . .	633
Figure 184. External trigger input block . . . . .	634
Figure 185. Control circuit in external clock mode 2 . . . . .	635
Figure 186. Capture/Compare channel (example: channel 1 input stage) . . . . .	635
Figure 187. Capture/Compare channel 1 main circuit . . . . .	636
Figure 188. Output stage of Capture/Compare channel (channel 1). . . . .	636
Figure 189. PWM input mode timing . . . . .	638
Figure 190. Output compare mode, toggle on OC1 . . . . .	640
Figure 191. Edge-aligned PWM waveforms (ARR=8) . . . . .	641
Figure 192. Center-aligned PWM waveforms (ARR=8) . . . . .	643
Figure 193. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . . .	644
Figure 194. Combined PWM mode on channels 1 and 3 . . . . .	645
Figure 195. Clearing TIMx_OCxREF . . . . .	646
Figure 196. Example of one-pulse mode. . . . .	647
Figure 197. Retriggerable one-pulse mode . . . . .	649
Figure 198. Example of counter operation in encoder interface mode . . . . .	650
Figure 199. Example of encoder interface mode with TI1FP1 polarity inverted . . . . .	651
Figure 200. Control circuit in reset mode . . . . .	652
Figure 201. Control circuit in gated mode . . . . .	653
Figure 202. Control circuit in trigger mode . . . . .	654
Figure 203. Control circuit in external clock mode 2 + trigger mode . . . . .	655
Figure 204. Master/Slave timer example . . . . .	656

Figure 205. Master/slave connection example with 1 channel only timers	656
Figure 206. Gating TIM2 with OC1REF of TIM3	657
Figure 207. Gating TIM2 with Enable of TIM3	658
Figure 208. Triggering TIM2 with update of TIM3	659
Figure 209. Triggering TIM2 with Enable of TIM3	659
Figure 210. Triggering TIM3 and TIM2 with TIM3 TI1 input.	660
Figure 211. Basic timer block diagram.	694
Figure 212. Counter timing diagram with prescaler division change from 1 to 2	696
Figure 213. Counter timing diagram with prescaler division change from 1 to 4	696
Figure 214. Counter timing diagram, internal clock divided by 1	697
Figure 215. Counter timing diagram, internal clock divided by 2	698
Figure 216. Counter timing diagram, internal clock divided by 4	698
Figure 217. Counter timing diagram, internal clock divided by N	699
Figure 218. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded).	699
Figure 219. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded).	700
Figure 220. Control circuit in normal mode, internal clock divided by 1	701
Figure 221. General-purpose timer block diagram (TIM14).	708
Figure 222. Counter timing diagram with prescaler division change from 1 to 2	710
Figure 223. Counter timing diagram with prescaler division change from 1 to 4	710
Figure 224. Counter timing diagram, internal clock divided by 1	711
Figure 225. Counter timing diagram, internal clock divided by 2	712
Figure 226. Counter timing diagram, internal clock divided by 4	712
Figure 227. Counter timing diagram, internal clock divided by N	713
Figure 228. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded).	713
Figure 229. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded).	714
Figure 230. Control circuit in normal mode, internal clock divided by 1	715
Figure 231. Capture/compare channel (example: channel 1 input stage)	715
Figure 232. Capture/compare channel 1 main circuit	716
Figure 233. Output stage of capture/compare channel (channel 1).	716
Figure 234. Output compare mode, toggle on OC1.	719
Figure 235. Edge-aligned PWM waveforms (ARR=8)	720
Figure 236. TIM15 block diagram	735
Figure 237. TIM16/TIM17 block diagram	736
Figure 238. Counter timing diagram with prescaler division change from 1 to 2	738
Figure 239. Counter timing diagram with prescaler division change from 1 to 4	738
Figure 240. Counter timing diagram, internal clock divided by 1	740
Figure 241. Counter timing diagram, internal clock divided by 2	740
Figure 242. Counter timing diagram, internal clock divided by 4	741
Figure 243. Counter timing diagram, internal clock divided by N	741
Figure 244. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded).	742
Figure 245. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded).	742
Figure 246. Update rate examples depending on mode and TIMx_RCR register settings	744
Figure 247. Control circuit in normal mode, internal clock divided by 1	745
Figure 248. TI2 external clock connection example.	745
Figure 249. Control circuit in external clock mode 1	746
Figure 250. Capture/compare channel (example: channel 1 input stage)	747



Figure 251. Capture/compare channel 1 main circuit	747
Figure 252. Output stage of capture/compare channel (channel 1)	748
Figure 253. Output stage of capture/compare channel (channel 2 for TIM15)	748
Figure 254. PWM input mode timing	750
Figure 255. Output compare mode, toggle on OC1	752
Figure 256. Edge-aligned PWM waveforms (ARR=8)	753
Figure 257. Combined PWM mode on channel 1 and 2	754
Figure 258. Complementary output with dead-time insertion	755
Figure 259. Dead-time waveforms with delay greater than the negative pulse	755
Figure 260. Dead-time waveforms with delay greater than the positive pulse	756
Figure 261. Break circuitry overview	758
Figure 262. Output behavior in response to a break	760
Figure 263. Output redirection	762
Figure 264. 6-step generation, COM example (OSSR=1)	763
Figure 265. Example of one pulse mode	764
Figure 266. Retriggerable one pulse mode	766
Figure 267. Measuring time interval between edges on 2 signals	767
Figure 268. Control circuit in reset mode	768
Figure 269. Control circuit in gated mode	769
Figure 270. Control circuit in trigger mode	770
Figure 271. Low-power timer block diagram (LPTIM1 and LPTIM2 <sup>(1)</sup> )	824
Figure 272. Glitch filter timing diagram	828
Figure 273. LPTIM output waveform, single counting mode configuration	829
Figure 274. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)	830
Figure 275. LPTIM output waveform, Continuous counting mode configuration	830
Figure 276. Waveform generation	832
Figure 277. Encoder mode counting sequence	836
Figure 278. IRTIM internal hardware connections	849
Figure 279. Independent watchdog block diagram	850
Figure 280. Watchdog block diagram	860
Figure 281. Window watchdog timing diagram	861
Figure 282. RTC block diagram	866
Figure 283. TAMP block diagram	905
Figure 284. Block diagram	921
Figure 285. I <sup>2</sup> C-bus protocol	923
Figure 286. Setup and hold timings	925
Figure 287. I2C initialization flow	927
Figure 288. Data reception	928
Figure 289. Data transmission	929
Figure 290. Target initialization flow	932
Figure 291. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0	934
Figure 292. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1	935
Figure 293. Transfer bus diagrams for I2C target transmitter (mandatory events only)	936
Figure 294. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0	937
Figure 295. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1	938
Figure 296. Transfer bus diagrams for I2C target receiver (mandatory events only)	938
Figure 297. Controller clock generation	940
Figure 298. Controller initialization flow	942
Figure 299. 10-bit address read access with HEAD10R = 0	942
Figure 300. 10-bit address read access with HEAD10R = 1	943

Figure 301. Transfer sequence flow for I2C controller transmitter, $N \leq 255$ bytes. ....	944
Figure 302. Transfer sequence flow for I2C controller transmitter, $N > 255$ bytes. ....	945
Figure 303. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . . .	946
Figure 304. Transfer sequence flow for I2C controller receiver, $N \leq 255$ bytes . . . . .	948
Figure 305. Transfer sequence flow for I2C controller receiver, $N > 255$ bytes. ....	949
Figure 306. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . . .	950
Figure 307. Timeout intervals for $t_{LOW:SEXT}$ , $t_{LOW:MEXT}$ . . . . .	955
Figure 308. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . . .	958
Figure 309. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . . .	959
Figure 310. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . . .	960
Figure 311. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . . .	961
Figure 312. Bus transfer diagrams for SMBus controller transmitter . . . . .	962
Figure 313. Bus transfer diagrams for SMBus controller receiver . . . . .	964
Figure 314. USART block diagram . . . . .	988
Figure 315. Word length programming . . . . .	991
Figure 316. Configurable stop bits . . . . .	993
Figure 317. TC/TXE behavior when transmitting . . . . .	996
Figure 318. Start bit detection when oversampling by 16 or 8 . . . . .	997
Figure 319. usart_ker_ck clock divider block diagram . . . . .	1000
Figure 320. Data sampling when oversampling by 16 . . . . .	1001
Figure 321. Data sampling when oversampling by 8 . . . . .	1002
Figure 322. Mute mode using Idle line detection . . . . .	1009
Figure 323. Mute mode using address mark detection . . . . .	1010
Figure 324. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . . .	1013
Figure 325. Break detection in LIN mode vs. Framing error detection . . . . .	1014
Figure 326. USART example of synchronous master transmission . . . . .	1015
Figure 327. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . . .	1015
Figure 328. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . . .	1016
Figure 329. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . . .	1017
Figure 330. ISO 7816-3 asynchronous protocol . . . . .	1019
Figure 331. Parity error detection using the 1.5 stop bits . . . . .	1021
Figure 332. IrDA SIR ENDEC block diagram . . . . .	1025
Figure 333. IrDA data modulation (3/16) - normal mode . . . . .	1025
Figure 334. Transmission using DMA . . . . .	1027
Figure 335. Reception using DMA . . . . .	1028
Figure 336. Hardware flow control between 2 USARTs . . . . .	1028
Figure 337. RS232 RTS flow control . . . . .	1029
Figure 338. RS232 CTS flow control . . . . .	1030
Figure 339. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . . .	1033
Figure 340. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . . .	1033
Figure 341. LPUART block diagram . . . . .	1074
Figure 342. LPUART word length programming . . . . .	1077
Figure 343. Configurable stop bits . . . . .	1079
Figure 344. TC/TXE behavior when transmitting . . . . .	1081
Figure 345. lpuart_ker_ck clock divider block diagram . . . . .	1084
Figure 346. Mute mode using Idle line detection . . . . .	1088



Figure 347. Mute mode using address mark detection	1089
Figure 348. Transmission using DMA	1091
Figure 349. Reception using DMA	1092
Figure 350. Hardware flow control between 2 LPUARTs	1093
Figure 351. RS232 RTS flow control	1093
Figure 352. RS232 CTS flow control	1094
Figure 353. Wake-up event verified (wake-up event = address match, FIFO disabled)	1097
Figure 354. Wake-up event not verified (wake-up event = address match, FIFO disabled)	1097
Figure 355. SPI block diagram	1126
Figure 356. Full-duplex single master/ single slave application	1127
Figure 357. Half-duplex single master/ single slave application	1128
Figure 358. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)	1129
Figure 359. Master and three independent slaves	1130
Figure 360. Multimaster application	1131
Figure 361. Hardware/software slave select management	1132
Figure 362. Data clock timing diagram	1133
Figure 363. Data alignment when data length is not equal to 8-bit or 16-bit	1134
Figure 364. Packing data in FIFO for transmission and reception	1138
Figure 365. Master full-duplex communication	1141
Figure 366. Slave full-duplex communication	1142
Figure 367. Master full-duplex communication with CRC	1143
Figure 368. Master full-duplex communication in packed mode	1144
Figure 369. NSSP pulse generation in Motorola SPI master mode	1147
Figure 370. TI mode transfer	1148
Figure 371. I2S block diagram	1151
Figure 372. I <sup>2</sup> S Philips protocol waveforms (16/32-bit full accuracy)	1153
Figure 373. I <sup>2</sup> S Philips standard waveforms (24-bit frame)	1153
Figure 374. Transmitting 0x8EAA33	1154
Figure 375. Receiving 0x8EAA33	1154
Figure 376. I <sup>2</sup> S Philips standard (16-bit extended to 32-bit packet frame)	1154
Figure 377. Example of 16-bit data frame extended to 32-bit channel frame	1154
Figure 378. MSB Justified 16-bit or 32-bit full-accuracy length	1155
Figure 379. MSB justified 24-bit frame length	1155
Figure 380. MSB justified 16-bit extended to 32-bit packet frame	1156
Figure 381. LSB justified 16-bit or 32-bit full-accuracy	1156
Figure 382. LSB justified 24-bit frame length	1156
Figure 383. Operations required to transmit 0x3478AE	1157
Figure 384. Operations required to receive 0x3478AE	1157
Figure 385. LSB justified 16-bit extended to 32-bit packet frame	1157
Figure 386. Example of 16-bit data frame extended to 32-bit channel frame	1158
Figure 387. PCM standard waveforms (16-bit)	1158
Figure 388. PCM standard waveforms (16-bit extended to 32-bit packet frame)	1159
Figure 389. Start sequence in master mode	1160
Figure 390. Audio sampling frequency definition	1161
Figure 391. I <sup>2</sup> S clock generator architecture	1161
Figure 392. CAN subsystem	1183
Figure 393. FDCAN block diagram	1185
Figure 394. Bit timing	1187
Figure 395. Transceiver delay measurement	1192

Figure 396. Pin control in bus monitoring mode . . . . .	1193
Figure 397. Pin control in loop-back mode . . . . .	1196
Figure 398. CAN error state diagram . . . . .	1197
Figure 399. Message RAM configuration . . . . .	1198
Figure 400. Standard message ID filter path . . . . .	1201
Figure 401. Extended message ID filter path . . . . .	1202
Figure 402. USB peripheral block diagram . . . . .	1249
Figure 403. Packet buffer areas with examples of buffer description table locations . . . . .	1256
Figure 404. UCPD block diagram . . . . .	1298
Figure 405. Clock division and timing elements . . . . .	1299
Figure 406. K-code transmission . . . . .	1302
Figure 407. Transmit order for various sizes of data . . . . .	1303
Figure 408. Packet format . . . . .	1304
Figure 409. Line format of Hard Reset . . . . .	1304
Figure 410. Line format of Cable Reset . . . . .	1305
Figure 411. BIST test data frame . . . . .	1306
Figure 412. BIST Carrier Mode 2 frame . . . . .	1306
Figure 413. UCPD BMC transmitter architecture . . . . .	1307
Figure 414. UCPD BMC receiver architecture . . . . .	1308
Figure 415. HDMI-CEC block diagram . . . . .	1336
Figure 416. Message structure . . . . .	1337
Figure 417. Blocks . . . . .	1337
Figure 418. Bit timings . . . . .	1338
Figure 419. Signal free time . . . . .	1338
Figure 420. Arbitration phase . . . . .	1339
Figure 421. SFT of three nominal bit periods . . . . .	1339
Figure 422. Error bit timing . . . . .	1340
Figure 423. Error handling . . . . .	1341
Figure 424. TXERR detection . . . . .	1343
Figure 425. Block diagram of STM32G0x1 MCU and Cortex <sup>®</sup> -M0+-level debug support . . . . .	1353

# 1 Documentation conventions

## 1.1 General information

The STM32G0x1 devices have an Arm<sup>®(a)</sup> Cortex<sup>®</sup>-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<sup>®</sup>-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:** device 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 devices, refer to the particular device datasheet.

The following table shows per-device availability of peripherals that are not common to all STM32G0x1 devices. The “X” means that the peripheral is available, “-” not available.

**Table 1. Peripherals versus devices**

Feature	STM32G0 31	STM32G0 41	STM32G0 51	STM32G0 61	STM32G0 71	STM32G0 81	STM32G0 B1	STM32G0 C1
CRS	-	-	-	-	-	-	X	X
RNG	-	X	-	X	-	X	-	X
AES	-	X	-	X	-	X	-	X
DAC	-	-	X	X	X	X	X	X
COMP1, COMP2	-	-	X	X	X	X	X	X
COMP3	-	-	-	-	-	-	X	X
TIM4	-	-	-	-	-	-	X	X
TIM6 and TIM7	-	-	X	X	X	X	X	X
TIM15	-	-	X	X	X	X	X	X
I2C2	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(2)</sup>	X <sup>(2)</sup>
I2C3	-	-	-	-	-	-	X <sup>(1)</sup>	X <sup>(1)</sup>
SPI3	-	-	-	-	-	-	X	X
I2S2	-	-	-	-	-	-	X	X

Table 1. Peripherals versus devices (continued)

Feature	STM32G031	STM32G041	STM32G051	STM32G061	STM32G071	STM32G081	STM32G0B1	STM32G0C1
USART2	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(2)</sup>	X <sup>(2)</sup>	X <sup>(2)</sup>	X <sup>(2)</sup>
USART3	-	-	-	-	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(2)</sup>	X <sup>(2)</sup>
USART4	-	-	-	-	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>	X <sup>(1)</sup>
USART5, USART6	-	-	-	-	-	-	X <sup>(1)</sup>	X <sup>(1)</sup>
LPUART2	-	-	-	-	-	-	X	X
USB	-	-	-	-	-	-	X	X
UCPD1, UCPD2	-	-	-	-	X	X	X	X
FDCAN1, FDCAN2	-	-	-	-	-	-	X	X
CEC	-	-	-	-	X	X	X	X
DMA2	-	-	-	-	-	-	X	X
MCO2	-	-	-	-	-	-	X	X
HSI48 RC	-	-	-	-	-	-	X	X
CRS	-	-	-	-	-	-	X	X
GPIO port E	-	-	-	-	-	-	X	X
Switchable I/O clamping diode	X	X	X	X	-	-	X	X
V <sub>DDIO2</sub> monitor	-	-	-	-	-	-	X	X

1. Basic configuration
2. Full configuration

## 2 Memory and bus architecture

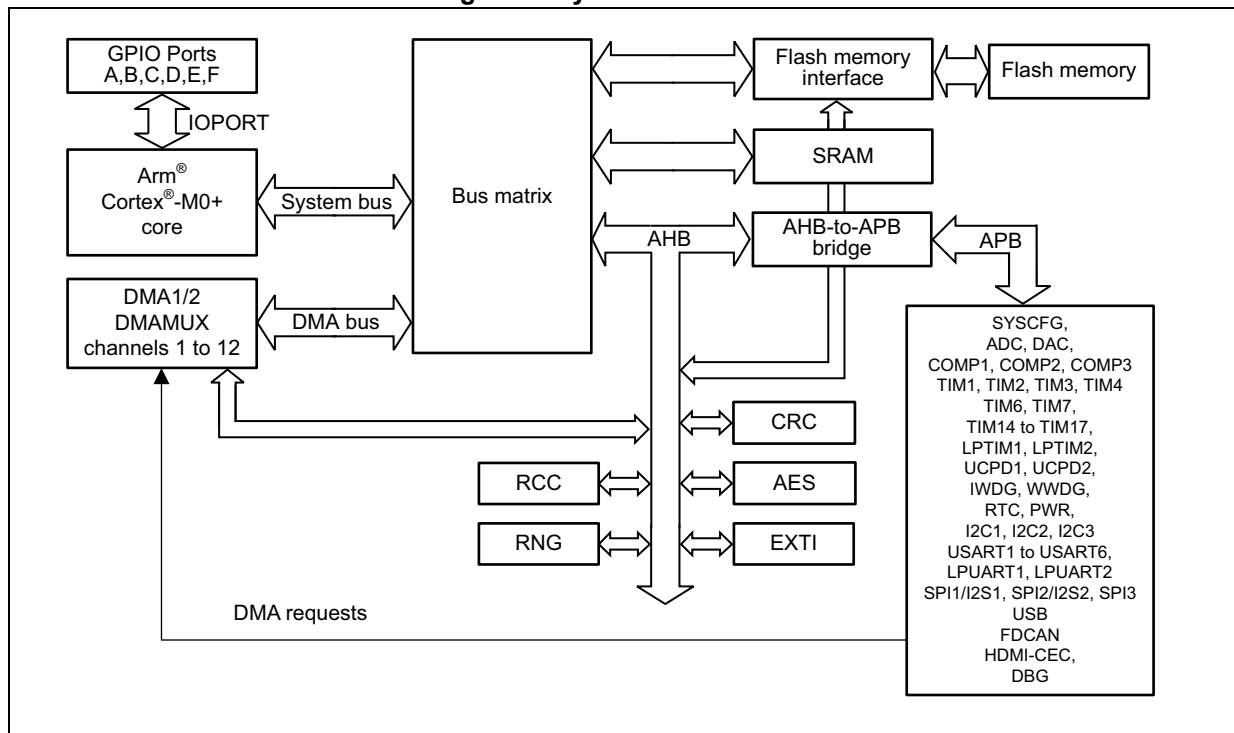
### 2.1 System architecture

The main system consists of:

- Two masters:
  - Cortex<sup>®</sup>-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<sup>®</sup>-M0+ core (peripheral bus) to a bus matrix that manages the arbitration between the core and the DMA.

#### DMA bus

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

### Bus matrix

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

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

### AHB-to-APB bridge (APB)

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

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

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

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

## **2.2 Memory organization**

### **2.2.1 Introduction**

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

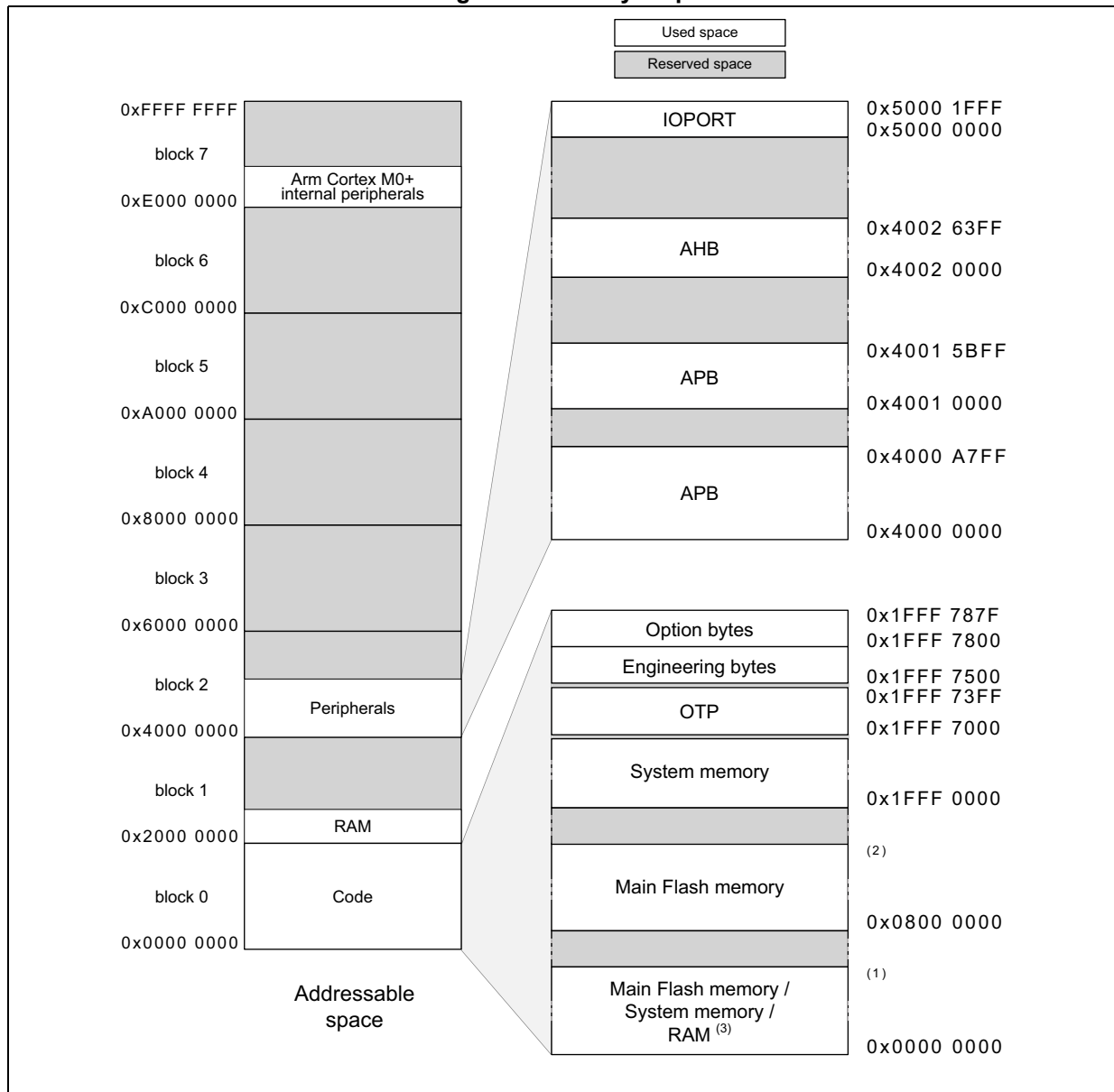
The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

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



## 2.2.2 Memory map and register boundary addresses

Figure 2. Memory map



1. STM32G0B1xx and STM32G0C1xx: 0x0007 FFFF; STM32G071xx and STM32G081xx: 0x0001 FFFF; STM32G051xx and STM32G061xx, STM32G031xx and STM32G041xx: 0x0000 FFFF.
2. STM32G0B1xx and STM32G0C1xx: 0x0807 FFFF; STM32G071xx and STM32G081xx: 0x0801 FFFF; STM32G051xx and STM32G061xx, STM32G031xx and STM32G041xx: 0x0800 FFFF.
3. 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 2. STM32G0B1xx and STM32G0C1xx memory boundary addresses**

Type	Boundary address	Size	Memory Area	Register description
SRAM	0x2002 4000 - 0x3FFF FFFF	~512 MB	Reserved	-
	0x2000 0000 - 0x2002 3FFF	144 KB	SRAM	<a href="#">Section 2.3 on page 66</a>
Code	0x1FFF 7880- 0x1FFF FFFF	~34 KB	Reserved	-
	0x1FFF 7800 - 0x1FFF 787F	128 B	Option bytes	<a href="#">Section 3.4 on page 82</a>
	0x1FFF 7500 - 0x1FFF 77FF	768 B	Engineering bytes	-
	0x1FFF 7400- 0x1FFF 74FF	256 B	Reserved	-
	0x1FFF 7000 - 0x1FFF 73FF	1 KB	OTP	-
	0x1FFF 0000 - 0x1FFF 6FFF	28 KB	System memory	-
	0x0808 0000 - 0x1FFF D7FF	~384 MB	Reserved	-
	0x0800 0000 - 0x0807 FFFF	512 KB	Main flash memory	<a href="#">Section 3.3.1 on page 71</a>
	0x0008 0000 - 0x07FF FFFF	~128 MB	Reserved	-
	0x0000 0000 - 0x0007 FFFF	512 KB	Main flash memory, system memory or SRAM depending on boot configuration	-

**Table 3. STM32G071xx and STM32G081xx memory boundary addresses**

Type	Boundary address	Size	Memory Area	Register description
SRAM	0x2000 9000 - 0x3FFF FFFF	~512 MB	Reserved	-
	0x2000 0000 - 0x2000 8FFF	36 KB	SRAM	<a href="#">Section 2.3 on page 66</a>
Code	0x1FFF 7880- 0x1FFF FFFF	~34 KB	Reserved	-
	0x1FFF 7800 - 0x1FFF 787F	128 B	Option bytes	<a href="#">Section 3.4 on page 82</a>
	0x1FFF 7500 - 0x1FFF 77FF	768 B	Engineering bytes	-
	0x1FFF 7400- 0x1FFF 74FF	256 B	Reserved	-
	0x1FFF 7000 - 0x1FFF 73FF	1 KB	OTP	-
	0x1FFF 0000 - 0x1FFF 6FFF	28 KB	System memory	-
	0x0802 0000 - 0x1FFF D7FF	~384 MB	Reserved	-
	0x0800 0000 - 0x0801 FFFF	128 KB	Main flash memory	<a href="#">Section 3.3.1 on page 71</a>
	0x0002 0000 - 0x07FF FFFF	~128 MB	Reserved	-
	0x0000 0000 - 0x0001 FFFF	128 KB	Main flash memory, system memory or SRAM depending on boot configuration	-

**Table 4. STM32G051xx and STM32G061xx memory boundary addresses**

Type	Boundary address	Size	Memory Area	Register description
SRAM	0x2000 4800 - 0x3FFF FFFF	~512 MB	Reserved	-
	0x2000 0000 - 0x2000 47FF	18 KB	SRAM	<a href="#">Section 2.3 on page 66</a>

**Table 4. STM32G051xx and STM32G061xx memory boundary addresses (continued)**

Type	Boundary address	Size	Memory Area	Register description
Code	0x1FFF 7880- 0x1FFF FFFF	~34 KB	Reserved	-
	0x1FFF 7800 - 0x1FFF 787F	128 B	Option bytes	<a href="#">Section 3.4 on page 82</a>
	0x1FFF 7500 - 0x1FFF 77FF	768 B	Engineering bytes	-
	0x1FFF 7400- 0x1FFF 74FF	256 B	Reserved	-
	0x1FFF 7000 - 0x1FFF 73FF	1 KB	OTP	-
	0x1FFF 2000 - 0x1FFF 6FFF	~20 KB	Reserved	-
	0x1FFF 0000 - 0x1FFF 1FFF	8 KB	System memory	-
	0x0801 0000 - 0x1FFF D7FF	~384 MB	Reserved	-
	0x0800 0000 - 0x0800 FFFF	64 KB	Main flash memory	<a href="#">Section 3.3.1 on page 71</a>
	0x0001 0000 - 0x07FF FFFF	~128 MB	Reserved	-
	0x0000 0000 - 0x0000 FFFF	64 KB	Main flash memory, system memory or SRAM depending on boot configuration	-

**Table 5. STM32G031xx and STM32G041xx memory boundary addresses**

Type	Boundary address	Size	Memory Area	Register description
SRAM	0x2000 2000 - 0x3FFF FFFF	~512 MB	Reserved	-
	0x2000 0000 - 0x2000 1FFF	8 KB	SRAM	<a href="#">Section 2.3 on page 66</a>
Code	0x1FFF 7880- 0x1FFF FFFF	~34 KB	Reserved	-
	0x1FFF 7800 - 0x1FFF 787F	128 B	Option bytes	<a href="#">Section 3.4 on page 82</a>
	0x1FFF 7500 - 0x1FFF 77FF	768 B	Engineering bytes	-
	0x1FFF 7400- 0x1FFF 74FF	256 B	Reserved	-
	0x1FFF 7000 - 0x1FFF 73FF	1 KB	OTP	-
	0x1FFF 2000 - 0x1FFF 6FFF	~20 KB	Reserved	-
	0x1FFF 0000 - 0x1FFF 1FFF	8 KB	System memory	-
	0x0801 0000 - 0x1FFF D7FF	~384 MB	Reserved	-
	0x0800 0000 - 0x0800 FFFF	64 KB	Main flash memory	<a href="#">Section 3.3.1 on page 71</a>
	0x0001 0000 - 0x07FF FFFF	~128 MB	Reserved	-
	0x0000 0000 - 0x0000 FFFF	64 KB	Main flash memory, system memory or SRAM depending on boot configuration	-

The following table gives the boundary addresses of the peripherals.

**Table 6. STM32G0x1 peripheral register boundary addresses**

Bus	Boundary address	Size	Peripheral	Peripheral register map
-	0xE000 0000 - 0xE00F FFFF	1MB	Cortex <sup>®</sup> -M0+ internal peripherals	-
IOPORT	0x5000 1800 - 0x5FFF FFFF	~256 MB	Reserved	-
	0x5000 1400 - 0x5000 17FF	1 KB	GPIOF	<a href="#">Section 7.5.12 on page 245</a>
	0x5000 1000 - 0x5000 13FF	1 KB	GPIOE	<a href="#">Section 7.5.12 on page 245</a>
	0x5000 0C00 - 0x5000 0FFF	1 KB	GIOD	<a href="#">Section 7.5.12 on page 245</a>
	0x5000 0800 - 0x5000 0BFF	1 KB	GPIOC	<a href="#">Section 7.5.12 on page 245</a>
	0x5000 0400 - 0x5000 07FF	1 KB	GPIOB	<a href="#">Section 7.5.12 on page 245</a>
	0x5000 0000 - 0x5000 03FF	1 KB	GPIOA	<a href="#">Section 7.5.12 on page 245</a>
AHB	0x4002 6400 - 0x4FFF FFFF	~256 MB	Reserved	-
	0x4002 6000 - 0x4002 63FF	1 KB	AES	<a href="#">Section 20.7.18 on page 513</a>
	0x4002 5400 - 0x4002 5FFF	3 KB	Reserved	-
	0x4002 5000 - 0x4002 53FF	1 KB	RNG	<a href="#">Section 19.7.4 on page 465</a>
	0x4002 3400 - 0x4002 4FFF	3 KB	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 KB	CRC	<a href="#">Section 14.4.6 on page 340</a>
	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 KB	FLASH	<a href="#">Section 3.7.22 on page 112</a>
	0x4002 1C00 - 0x4002 1FFF	3 KB	Reserved	-
	0x4002 1800 - 0x4002 1BFF	1 KB	EXTI	<a href="#">Section 13.5.16 on page 332</a>
	0x4002 1400 - 0x4002 17FF	1 KB	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 KB	RCC	<a href="#">Section 5.4.25 on page 213</a>
	0x4002 0C00 - 0x4002 0FFF	1 KB	Reserved	-
	0x4002 0800 - 0x4002 0BFF	1 KB	DMAMUX	<a href="#">Section 11.6.7 on page 310</a>
	0x4002 0400 - 0x4002 07FF	1 KB	DMA2	<a href="#">Section 10.6.7 on page 293</a>
	0x4002 0000 - 0x4002 03FF	1 KB	DMA1	<a href="#">Section 10.6.7 on page 293</a>
APB	0x4001 5C00 - 0x4001 FFFF	32 KB	Reserved	-
	0x4001 5800 - 0x4001 5BFF	1 KB	DBG	<a href="#">Section 40.10.5 on page 1366</a>
	0x4001 4C00 - 0x4001 57FF	3 KB	Reserved	-
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17	<a href="#">Section 25.6.21 on page 821</a>
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16	<a href="#">Section 25.6.21 on page 821</a>

Table 6. STM32G0x1 peripheral register boundary addresses (continued)

Bus	Boundary address	Size	Peripheral	Peripheral register map
APB	0x4001 4000 - 0x4001 43FF	1 KB	TIM15	<a href="#">Section 25.6.21 on page 821</a>
	0x4001 3C00 - 0x4001 3FFF	1 KB	USART6	<a href="#">Section 33.8.15 on page 1069</a>
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1	<a href="#">Section 33.8.15 on page 1069</a>
	0x4001 3400 - 0x4001 37FF	1 KB	Reserved	-
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1/I2S1	<a href="#">Section 35.9.10 on page 1181</a>
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1	<a href="#">Section 21.4 on page 576</a>
	0x4001 2800 - 0x4001 2BFF	1 KB	Reserved	-
	0x4001 2400 - 0x4001 27FF	1 KB	ADC	<a href="#">Section 15.13 on page 397</a>
	0x4001 0400 - 0x4001 23FF	8 KB	Reserved	-
	0x4001 0200 - 0x4001 03FF	1 KB	COMP	<a href="#">Section 18.6.4 on page 453</a>
	0x4001 0080 - 0x4001 01FF		SYSCFG(ITLINE) <sup>(1)</sup>	<a href="#">Section 8.1.35 on page 265</a>
	0x4001 0030 - 0x4001 007F		VREFBUF	<a href="#">Section 17.3.3 on page 439</a>
	0x4001 0000 - 0x4001 002F		SYSCFG	<a href="#">Section 8.1.35 on page 265</a>
	0x4000 BC00 - 0x4000 FFFF	17 KB	Reserved	-
	0x4000 B400 - 0x4000 BBFF	2 KB	FDCAN message RAM	-
	0x4000 B000 - 0x4000 B3FF	1 KB	TAMP (+ BKP registers)	<a href="#">Section 31.6.9 on page 918</a>
	0x4000 A800 - 0x4000 AFFF	2 KB	Reserved	-
	0x4000 A400 - 0x4000 A7FF	1 KB	UCPD2	<a href="#">Section 38.8.15 on page 1332</a>
	0x4000 A000 - 0x4000 A3FF	1 KB	UCPD1	<a href="#">Section 38.8.15 on page 1332</a>
	0x4000 9C00 - 0x4000 9FFF	1 KB	USB RAM2	-
	0x4000 9800 - 0x4000 9BFF	1 KB	USB RAM1	-
	0x4000 9400 - 0x4000 97FF	1 KB	LPTIM2	<a href="#">Section 26.7.10 on page 848</a>
	0x4000 8C00 - 0x4000 93FF	2 KB	Reserved	-
	0x4000 8800 - 0x4000 8BFF	1 KB	I2C3	<a href="#">Section 32.9.12 on page 983</a>
	0x4000 8400 - 0x4000 87FF	1 KB	LPUART2	<a href="#">Section 33.8.15 on page 1069</a>
	0x4000 8000 - 0x4000 83FF	1 KB	LPUART1	<a href="#">Section 33.8.15 on page 1069</a>
	0x4000 7C00 - 0x4000 7FFF	1 KB	LPTIM1	<a href="#">Section 26.7.10 on page 848</a>
	0x4000 7800 - 0x4000 7BFF	1 KB	CEC	<a href="#">Section 39.7.7 on page 1352</a>
	0x4000 7400 - 0x4000 77FF	1 KB	DAC	<a href="#">Section 16.7.21 on page 430</a>
	0x4000 7000 - 0x4000 73FF	1 KB	PWR	<a href="#">Section 4.4.20 on page 153</a>
	0x4000 6C00 - 0x4000 6FFF	1 KB	CRS	<a href="#">Section 6.7.5 on page 227</a>
	0x4000 6800 - 0x4000 6BFF	1 KB	FDCAN2	<a href="#">Section 36.4.38 on page 1244</a>
0x4000 6400 - 0x4000 67FF	1 KB	FDCAN1	<a href="#">Section 36.4.38 on page 1244</a>	
0x4000 6000 - 0x4000 63FF	1 KB	Reserved	-	
0x4000 5C00 - 0x4000 5FFF	1 KB	USB	<a href="#">Section 37.6.8 on page 1289</a>	

Table 6. STM32G0x1 peripheral register boundary addresses (continued)

Bus	Boundary address	Size	Peripheral	Peripheral register map
APB	0x4000 5800 - 0x4000 5BFF	1 KB	I2C2	<a href="#">Section 32.9.12 on page 983</a>
	0x4000 5400 - 0x4000 57FF	1 KB	I2C1	<a href="#">Section 32.9.12 on page 983</a>
	0x4000 5000 - 0x4000 53FF	1 KB	USART5	<a href="#">Section 33.8.15 on page 1069</a>
	0x4000 4C00 - 0x4000 4FFF	1 KB	USART4	<a href="#">Section 33.8.15 on page 1069</a>
	0x4000 4800 - 0x4000 4BFF	1 KB	USART3	<a href="#">Section 33.8.15 on page 1069</a>
	0x4000 4400 - 0x4000 47FF	1 KB	USART2	<a href="#">Section 33.8.15 on page 1069</a>
	0x4000 4000 - 0x4000 43FF	1 KB	Reserved	-
	0x4000 3C00 - 0x4000 3FFF	1 KB	SPI3	<a href="#">Section 35.9.10 on page 1181</a>
	0x4000 3800 - 0x4000 3BFF	1 KB	SPI2/I2S2	<a href="#">Section 35.9.10 on page 1181</a>
	0x4000 3400 - 0x4000 37FF	1 KB	Reserved	-
	0x4000 3000 - 0x4000 33FF	1 KB	IWDG	<a href="#">Section 28.4.6 on page 858</a>
	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG	<a href="#">Section 29.5.4 on page 864</a>
	0x4000 2800 - 0x4000 2BFF	1 KB	RTC	<a href="#">Section 30.6.21 on page 902</a>
	0x4000 2400 - 0x4000 27FF	1 KB	Reserved	-
	0x4000 2000 - 0x4000 23FF	1 KB	TIM14	<a href="#">Section 24.4.13 on page 731</a>
	0x4000 1800 - 0x4000 1FFF	2 KB	Reserved	-
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7	<a href="#">Section 23.4.9 on page 706</a>
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6	<a href="#">Section 23.4.9 on page 706</a>
	0x4000 0C00 - 0x4000 0FFF	1 KB	Reserved	-
	0x4000 0800 - 0x4000 0BFF	1 KB	TIM4	<a href="#">Section 22.4.31 on page 691</a>
0x4000 0400 - 0x4000 07FF	1 KB	TIM3	<a href="#">Section 22.4.31 on page 691</a>	
0x4000 0000 - 0x4000 03FF	1 KB	TIM2	<a href="#">Section 22.4.31 on page 691</a>	

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 7. SRAM size

Device	SRAM with parity enabled (Kbyte)	SRAM with parity disabled (Kbyte)
STM32G0B1xx and STM32G0C1xx	128	144
STM32G071xx and STM32G081xx	32	36
STM32G051xx and STM32G061xx	16	18
STM32G031xx and STM32G041xx	8	8

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. In addition, to get the SRAM parity error at the same cycle time that it is occurring, a bus error is generated (triggering a `HardFault` exception) together with the NMI. This avoids the corrupted data to be used by the application, but with the side effect of having both NMI and `HardFault` interrupts generated. The same error can also be linked to the `BRK_IN` Break input of `TIM1/15/16/17`, with the `SRAM_PARITY_LOCK` control bit in the [SYSCFG configuration register 2 \(SYSCFG\\_CFGR2\)](#). The SRAM Parity Error flag (`SRAM_PEF`) is available in the [SYSCFG configuration register 2 \(SYSCFG\\_CFGR2\)](#).

*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 bootloader code
  - OTP (one-time programmable) areaRefer to [Section 3: Embedded flash memory \(FLASH\)](#) for more details.

The flash memory 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 memory registers.

## 2.5 Boot configuration

In the STM32G0x1, three different boot modes can be selected through the BOOT0 pin, BOOT\_LOCK bit in FLASH\_SECR register, and boot configuration bits nBOOT1, BOOT\_SEL and nBOOT0 in the user option byte, as shown in the following table.

**Table 8. Boot modes**

Boot mode configuration					Selected boot area
BOOT_LOCK bit	nBOOT1 bit	BOOT0 pin	nBOOT_SEL bit	nBOOT0 bit	
0	x	0	0	x	Main flash memory <sup>(1)</sup>
0	1	1	0	x	System memory
0	0	1	0	x	Embedded SRAM
0	x	x	1	1	Main flash memory <sup>(1)</sup>
0	1	x	1	0	System memory
0	0	x	1	0	Embedded SRAM
1	x	x	x	x	Main flash memory forced

1. Boot forced to system memory when EMPTY bit of the *FLASH access control register (FLASH\_ACR)* is set. See the *Empty check* section.

The BOOT0 pin is sampled on the NRST (external reset) rising edge. Refer to the NRST (external reset) description in [Section 5.1.2: System reset](#) for further details about how the PF2-NRST pin mode impacts the BOOT0 sampling. The user option bits are loaded during the option byte load (OBL) process.

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

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

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

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

**Caution:** BOOT0 pin shares the same GPIO with serial wire clock (SWCLK) that is used by the debugger to connect with the device, based on the fact that these functionalities can be considered almost completely disjoint. Nevertheless, to ensure system robustness, the



STM32G0x1 devices provide an hardware mechanism to force BOOT0 low (boot from User flash memory) if a debugger access is detected (and BOOT0 information is taken from the pin), in order to use SWCLK clock for debugger serial communications and at the same time have a safe boot configuration for the device itself. This configuration is kept until next power-on following debugger access.

### 2.5.1 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 \(SYSCFG\\_CFGR1\)](#).

### 2.5.2 Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. It is used to reprogram the flash memory using one of the following serial interfaces:

- USART or I2C (applies to all devices)
- SPI (applies to STM32G071xx and STM32G081xx and to STM32G0B1xx and STM32G0C1xx)
- USB (DFU) or FDCAN (applies to STM32G0B1xx and STM32G0C1xx)

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

### 2.5.3 Forcing boot from main flash memory

The BOOT\_LOCK bit allows forcing a unique entry point in the main flash memory for boot, regardless of the other boot mode configuration bits or the empty check flag. For more details, refer to [Section 3.5.6: Forcing boot from main flash memory](#).

### 2.5.4 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 bootloader. This flag is checked when the boot configuration defines the main flash memory as the target boot area and the BOOT\_LOCK bit is not set. When the EMPTY flag is set, the device is considered empty and system memory (bootloader) is selected instead of the main flash memory as a boot area, to allow the user to program the device. Refer to AN2606 for more details concerning the bootloader and GPIO configuration in system memory boot mode (some of the GPIOs are reconfigured from the High-Z state).

The EMPTY flag is updated by hardware only during the loading of option bytes: it is set when the full 72-bit content (including ECC) of the address 0x0800 0000 is read as 0xFF FFFF FFFF FFFF FFFF, otherwise it is cleared. It means that, after programming of a virgin device, a power on reset or setting of OBL\_LAUNCH bit in FLASH\_CR register is required to clear the EMPTY flag (the system reset has no impact on this flag). Software can also modify the EMPTY flag directly in the FLASH\_ACR register.

*Note:* *If the device is programmed for the first time but the EMPTY flag is not updated, the device still selects the system memory as a boot area after a system reset.*

## 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

- Up to 512 Kbytes of flash memory (main memory):
  - up to 64 Kbytes for STM32G031xx and STM32G041xx / STM32G051xx and STM32G061xx
  - up to 128 Kbytes for STM32G071xx and STM32G081xx
  - up to 512 Kbytes for STM32G0B1xx and STM32G0C1xx
- Memory organization:
  - 1 bank (devices with up to 128 Kbytes of flash memory)
  - 2 banks (devices with more than 128 Kbytes of flash memory)
  - Page size: 2 Kbytes
  - Subpage size: 512 bytes
- 72-bit wide data read (64 bits plus 8 ECC bits)
- 72-bit wide data write (64 bits plus 8 ECC bits)
- Page erase (2 Kbytes), bank (single-bank) erase, and mass (all-bank) erase

Flash memory interface features:

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

### 3.3 FLASH functional description

#### 3.3.1 FLASH organization

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

The flash memory is organized as follows:

- A main memory block divided into one or two banks of up to 128 pages of 2 Kbytes each, with eight rows of 256 bytes in each page
- An information block containing:
  - System memory from which the CPU boots in system memory boot mode. The area is reserved and contains the bootloader used to reprogram the flash memory through one of the interfaces listed in the [Section 2.5.2: Embedded bootloader](#). 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. Once written, the entire double word cannot be written anymore, even with the value 0x0000 0000 0000 0000.
  - Option bytes for user configuration.

The following tables show the mapping of the flash memory into information block and main memory area.

**Table 9. Flash memory organization for single-bank devices**

Area	Addresses	Size (bytes)	16 KB devices	32 KB devices	64 KB devices	128 KB devices	256 KB devices (1)
Information block	0x1FFF 7800 - 0x1FFF 787F	128	Option bytes				
	0x1FFF 7500 - 0x1FFF 77FF	768	Engineering bytes				
	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area				
	0x1FFF 2000 - 0x1FFF 6FFF	20 K	-			System memory	
	0x1FFF 0000 - 0x1FFF 1FFF	8 K	System memory				

**Table 9. Flash memory organization for single-bank devices (continued)**

Area	Addresses	Size (bytes)	16 KB devices	32 KB devices	64 KB devices	128 KB devices	256 KB devices (1)	
Main memory (Bank 1)	0x0803 F800 - 0x0803 FFFF	2 K					-	Page 127
	...	...						...
	0x0801 F800 - 0x0801 FFFF	2 K						Page 63
	...	...						...
	0x0800 F800 - 0x0800 FFFF	2 K						Page 31
	...	...						...
	0x0800 7800 - 0x0800 7FFF	2 K						Page 15
	...	...						...
	0x0800 3800 - 0x0800 3FFF	2 K						Page 7
	...	...						...
	0x0800 0800 - 0x0800 0FFF	2 K						Page 1
0x0800 0000 - 0x0800 07FF	2 K						Page 0	

1. Flash memory configured in single-bank mode, by clearing the DUAL\_BANK option bit.

The following table shows the flash memory organization of the 256 KB device when its flash memory is configured in dual-bank mode, by setting the DUAL\_BANK option bit.

**Table 10. Flash memory organization for 256 KB dual-bank devices**

Area		Addresses	Size (bytes)	Memory type
Information block	Bank 1	0x1FFF 7800 - 0x1FFF 787F	128	Option bytes
		0x1FFF 7500 - 0x1FFF 77FF	768	Engineering bytes
		0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
		0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
Main memory	Bank 2 <sup>(1)</sup>	0x0803 F800 - 0x0803 FFFF	2 K	Page 63
		...	...	...
		0x0802 0800 - 0x0802 0FFF	2 K	Page 1
	Bank 1 <sup>(1)</sup>	0x0802 0000 - 0x0802 07FF	2 K	Page 0
		0x0801 F800 - 0x0801 FFFF	2 K	Page 63
		...	...	...
		0x0800 0800 - 0x0800 0FFF	2 K	Page 1
0x0800 0000 - 0x0800 07FF	2 K	Page 0		

1. If the banks are swapped (see nSWAP\_BANK option bit description), the bank 2 is addressed as the memory range 0x0800 0000 - 0x0801 FFFF and the bank 1 as 0x0802 0000 - 0x0803 FFFF.

**Table 11. Flash memory organization for 512 KB devices**

Area		Addresses	Size (bytes)	Memory type
Information block	Bank 1	0x1FFF 7800 - 0x1FFF 787F	128	Option bytes
		0x1FFF 7500 - 0x1FFF 77FF	768	Engineering bytes
		0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
		0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
Main memory	Bank 2 <sup>(1)</sup>	0x0807 F800 - 0x0807 FFFF	2 K	Page 127
		...	...	...
		0x0804 0800 - 0x0804 0FFF	2 K	Page 1
	Bank 1 <sup>(1)</sup>	0x0804 0000 - 0x0804 07FF	2 K	Page 0
		0x0803 F800 - 0x0803 FFFF	2 K	Page 127
		...	...	...
		0x0800 0800 - 0x0800 0FFF	2 K	Page 1
		0x0800 0000 - 0x0800 07FF	2 K	Page 0

1. If the banks are swapped (see nSWAP\_BANK option bit description), the bank 2 is addressed as the memory range 0x0800 0000 - 0x0803 FFFF and the bank 1 as 0x0804 0000 - 0x0807 FFFF.

### 3.3.2 FLASH dual-bank capability

The STM32G0B1xx and STM32G0C1xx devices support dual-bank flash memory capability. This feature allows using the read-while-write (RWW) function. It is also possible to swap, by hardware, the logical address mapping of the physical flash memory banks. The dual-bank and the bank swap functions are controlled with the DUAL\_BANK and nSWAP\_BANK option bits, respectively.

**Table 12. Flash memory bank mapping**

Device flash size	DUAL_BANK	Flash mode	nSWAP_BANK	Address mapping
512 Kbytes	X <sup>(1)</sup>	Dual-bank	1	Bank 1: 0x0800 0000 – 0x0803 FFFF Bank 2: 0x0804 0000 – 0x0807 FFFF
			0	Bank 1: 0x0804 0000 – 0x0807 FFFF Bank 2: 0x0800 0000 – 0x0803 FFFF
256 Kbytes	1		1	Bank 1: 0x0800 0000 – 0x0801 FFFF Bank 2: 0x0802 0000 – 0x0803 FFFF
			0	Bank 1: 0x0802 0000 – 0x0803 FFFF Bank 2: 0x0800 0000 – 0x0801 FFFF
128 Kbytes and less	X <sup>(3)</sup> or N/A	Single-bank	X <sup>(2)</sup>	Bank 1: 0x0800 0000 – 0x0803 FFFF Bank 2: not available
			X <sup>(2)</sup> or N/A	Bank 1: 0x0800 0000 – 0x0801 FFFF Bank 2: not available

1. The 512 Kbyte device always operates in dual-bank mode.
2. nSWAP\_BANK has no effect when the device is configured in single-bank mode.
3. The 128 Kbyte device always operates in single-bank mode.

### Flash memory programming

The flash memory programming is always performed at a specified logical address in the flash memory range. The hardware automatically performs the programming sequence in the right bank following the remapping shown in [Table 12: Flash memory bank mapping](#). Consequently, no specific sequence is required, regardless of the flash memory mode and bank swapping used.

### Flash memory erasing

The flash memory erase operation can be performed at page level (page erase) or bank level (mass erase). As shown in [Section 3.3.7: FLASH main memory erase sequences](#), the erase operation is always linked to a specific physical flash memory bank (that is, bank 1 or bank 2). Consequently, the erase or mass erase operation is not affected by the nSWAP\_BANK option bit.

### Flash memory write protection

The flash memory write protection configuration in the option bytes is also linked with physical flash banks. Therefore, care must be taken to select WRP1 or WRP2, depending on the nSWAP\_BANK configuration.

## 3.3.3 FLASH error code correction (ECC)

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

- One error detection and correction
- Two errors detection

When one error is detected and corrected, the flag ECCC (ECC correction) is set in [FLASH ECC register \(FLASH\\_ECCR\)](#) or in [FLASH ECC register 2 \(FLASH\\_ECCR2\)](#) for error in bank 2. If ECCCIE is set, an interrupt is generated.

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

When an ECC error is detected, the address of the failing double word is saved in ADDR\_ECC[15:0] bitfield of the FLASH\_ECCR register. ADDR\_ECC[2:0] are always cleared.

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

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

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

*The ECC status for the bank 2 is reported in [FLASH ECC register 2 \(FLASH\\_ECCR2\)](#).*

### 3.3.4 FLASH read access latency

To correctly read data from flash memory, the number of wait states (LATENCY) must be correctly programmed in the *FLASH access control register (FLASH\_ACR)* according to the frequency of the flash memory clock (HCLK) and the internal voltage range of the device  $V_{CORE}$ . Refer to *Section 4.1.4: Dynamic voltage scaling management. Table 13* shows the correspondence between wait states and flash memory clock frequency.

**Table 13. Number of wait states according to flash memory clock (HCLK) frequency**

Wait states (WS) (LATENCY)	HCLK (MHz)	
	$V_{CORE}$ Range 1	$V_{CORE}$ Range 2
0 WS (1 HCLK cycles)	≤ 24	≤ 8
1 WS (2 HCLK cycles)	≤ 48	≤ 16
2 WS (3 HCLK cycles)	≤ 64	-

After power reset, the HCLK clock frequency is 16 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH\_ACR register.

When waking up from Standby, the HCLK clock frequency is 16 MHz in Range 1 and 0 wait state (WS) is configured in the FLASH\_ACR register.

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

#### Increasing the CPU frequency

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

#### Decreasing the CPU frequency

1. Modify the system clock source by writing the SW bits of the RCC\_CFGR register.
2. If needed, modify the core clock prescaler by writing the HPRE bits of RCC\_CFGR.
3. Check that the new system clock source or/and the new core clock prescaler value is/are taken into account by reading the clock source status (SWS bits) of the RCC\_CFGR register, or/and the AHB prescaler value (HPREF bit), of the RCC\_CFGR register.

register, and wait until the programmed new system clock source or/and new flash memory clock prescaler value is/are read.

4. Program the new number of wait states to the LATENCY bits of the *FLASH access control register (FLASH\_ACR)*.
5. Optionally, check that the new number of wait states is used to access the flash memory by reading back the LATENCY bits of the *FLASH access control register (FLASH\_ACR)*.

### 3.3.5 FLASH memory acceleration

#### Instruction prefetch

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

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

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

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

**Caution:** In rare cases, the instruction prefetch may fail upon transiting across flash memory banks. Therefore, it is mandatory to temporarily disable the instruction prefetch when the application code branches or does function calls from one flash memory bank to the other (in dual-bank flash memory configuration). Refer to the corresponding product errata sheet for more details on this limitation.

#### 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<sup>®</sup>-M0+.



### 3.3.6 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 bootloader, 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.

**Caution:** Due to the structure of the flash memory (64 bits of data with associated 8 bits of ECC), any ongoing program or erase memory operation interrupted by a physical phenomena such as reset or power-off might result in a corrupted flash memory data and/or ECC bits. The address concerned might trigger an ECC interrupt or an NMI when accessed later on. Consequently, it is important that the firmware always handles flash memory program and erase operations with care and that it manages ECC events.

During a program/erase operation in 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 register 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 register can be locked again by software, by setting its LOCK bit.

**Note:** *The FLASH\_CR register cannot be written when the BSY1 or the BSY2 bit of the FLASH status register (FLASH\_SR) is set. Any attempt to write to this register with the BSY1 or the BSY2 bit set causes the AHB bus to stall until both bits are cleared.*

### 3.3.7 FLASH main memory erase sequences

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

#### Flash memory page erase

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

Table 14. 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				
	Yes	Yes				
1	x		Protected pages only	No	Yes	

To erase a page (2 Kbytes), proceed as follows:

1. Check that no flash memory operation is ongoing on the targeted bank, by checking the BSY1 or BSY2 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. Check that CFGBSY is cleared.
4. Set the PER bit and select the page to erase (BKER, PNB) in the *FLASH control register (FLASH\_CR)*.
5. Set the STRT bit of the *FLASH control register (FLASH\_CR)*.
6. Wait until the CFGBSY bit of the *FLASH status register (FLASH\_SR)* is cleared.

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

**Flash memory 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 15. Mass erase overview

SEC_PROT	PCROP	WRP	PCROP_RDP	Comment	WRPERR	CPU bus error
0	No	No	x	Memory is erased	No	No
	No	Yes		Erase aborted (no erase started)	Yes	
	Yes	No				
	Yes	Yes				
1	x		Erase aborted (no erase started)	No	Yes	

To perform a mass erase, proceed as follows:

1. Check that no flash memory operation is ongoing on the targeted bank, by checking the BSY1 or BSY2 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. Check that CFGBSY bit of the *FLASH status register (FLASH\_SR)* is cleared.
4. Set the MER1 (for the bank 1 or single-bank device) and/or MER2 (for the bank 2) bit of the *FLASH control register (FLASH\_CR)*.



5. Set the STRT bit of the *FLASH control register (FLASH\_CR)*.
6. Wait until the CFGBSY bit of the *FLASH status register (FLASH\_SR)* is cleared.

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

### 3.3.8 FLASH main memory programming sequences

The flash memory is programmed 72 bits (64-bit data plus 8-bit ECC) at a time.

Programming 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

To program the flash memory in standard mode, proceed as follows:

1. Check that no main flash memory operation is ongoing on the targeted bank, by checking the BSY1 or BSY2 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. Check that CFGBSY bit of the *FLASH status register (FLASH\_SR)* is cleared.
4. Set the PG bit of the *FLASH control register (FLASH\_CR)*.
5. Perform the data write operation at the desired memory address, inside main memory block or OTP area. Only double word (64 bits) can be programmed.
  - a) Write a first word in an address aligned with double word
  - b) Write the second word.
6. Wait until the CFGBSY bit of the *FLASH status register (FLASH\_SR)* is cleared.
7. Check that EOP flag of the *FLASH status register (FLASH\_SR)* is set (programming operation succeeded), and clear it by software.
8. Clear the PG bit of the *FLASH control register (FLASH\_CR)* if there no more programming request anymore.

*Note:* When the flash memory interface has received a good sequence (a double word), programming is automatically launched and BSY1 or BSY2 bit is set. The internal oscillator HSI16 (16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC\_CR register.

ECC is calculated from the double word to program.

#### Fast programming

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

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

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

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

To program the main flash memory in fast programming mode, proceed as follows:

1. Perform a mass or page erase. If not, PGSERR is set.
2. Check that no main flash memory operation is ongoing on the targeted bank, by checking the BSY1 or BSY2 bit of the *FLASH status register (FLASH\_SR)*.
3. Check and clear all error programming flag due to a previous programming.
4. Check that the CFGBSY bit of the *FLASH status register (FLASH\_SR)* is cleared.
5. Set the FSTPG bit in *FLASH control register (FLASH\_CR)*.
6. Write 32 double words to program a row (256 bytes).
7. Wait until the CFGBSY bit of the *FLASH status register (FLASH\_SR)* is cleared.
8. Check that EOP flag of the *FLASH status register (FLASH\_SR)* is set (programming operation succeeded), and clear it by software.
9. Clear the FSTPG bit of the *FLASH status register (FLASH\_SR)* if there are no more programming requests anymore.

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

*When the flash memory interface has received the first double word, programming is automatically launched. The BSY1 or BSY2 bit is set when the high voltage is applied for the first double word, and it is cleared when the last double word has been programmed or in case of error. The internal oscillator HSI16 (16 MHz) is enabled automatically when FSTPG bit is set, and disabled automatically when FSTPG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC\_CR register.*

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

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

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

### Programming errors

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

- **PROGERR:** Programming Error  
In standard programming: PROGERR is set if the word to write is not previously erased (except if the value to program is full zero 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 does not belong to the same double word address.
- In fast programming: the data to program does not belong to the same row than the previous programmed double words, the address to program is not greater than the previous one, or the address to program is located in a region that cannot be programmed in Fast programming mode (such as the option bytes).

- **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, MER2, 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 or MER2 is set.
- In the page erase sequence: PG, FSTPG, MER1, and MER2 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.
- Attempt to write in Fast programming mode an address located in a region that cannot be programmed in this mode (such as the option bytes).

- **MISSERR:** Fast Programming Data Miss Error

In fast programming: all the data must be written successively. MISSERR is set if the previous data programming is finished and the next data to program is not written yet.

- **FASTERR:** Fast Programming Error

In fast programming: FASTERR is set if one of the following conditions occurs:

- when FSTPG bit is set for more than 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).

### 3.3.9 Read-while-write (RWW) function

Dual-bank devices support read-while-write function that allows reading from one bank while erasing or programming within the other bank.

*Note:* Write-while-write operation, such as erasing within one bank while programming the other, is not allowed.

## 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 16](#).

**Table 16. Option byte format**

63-56	55-48	47-40	39-32	31-24	23-16	15 -8	7-0
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 17](#) shows the organization of the option bytes (the lower word only) in the flash memory information block, as a superset for single-bank and dual-bank devices. The software can read the option bytes from these flash memory locations or from their corresponding option registers referenced in the table. The option bytes can be read from the flash memory locations listed in the table. Refer to sections [3.7.8](#) to [3.7.21](#) for the description of the option register bitfields, also applicable to the option byte bitfields.

Table 17. Organization of option bytes

Address <sup>(1)</sup>	Corresponding option register (section)	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					
0x1FFF7800	FLASH_OPTR (3.7.8)	Res.		IRHEN	NRST_MODE	nBOOT0	nBOOT1	nBOOT_SEL	Res.	RAM_PARITY_CHECK	DUAL_BANK	nSWAP_BANK	WWDG_SW	IWDG_STBY	IWDG_STOP	IWDG_SW	nRST_SHDW	nRST_STDBY	nRST_STOP	BORF_LEV	BORR_LEV	BOR_EN	RDP															
	Factory value	X	X	1	1	1	1	1	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	0			
0x1FFF7808	FLASH_PCROP1ASR (3.7.9)	Res.																								PCROP1A_STRT												
	Factory value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1			
0x1FFF7810	FLASH_PCROP1AER (3.7.10)	PCROP_RDP	Res.																								PCROP1A_END											
	Factory value	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0			
0x1FFF7818	FLASH_WRP1AR (3.7.11)	Res.										WRP1A_END				Res.						WRP1A_STRT																
	Factory value	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1			
0x1FFF7820	FLASH_WRP1BR (3.7.12)	Res.										WRP1B_END				Res.						WRP1B_STRT																
	Factory value	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1			
0x1FFF7828	FLASH_PCROP1BSR (3.7.13)	Res.																								PCROP1B_STRT												
	Factory value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1			
0x1FFF7830	FLASH_PCROP1BER (3.7.14)	Res.																								PCROP1B_END												
	Factory value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0			
0x1FFF7838	FLASH_PCROP1ASR (3.7.15)	Res.																								PCROP2A_STRT												
	Factory value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1			
0x1FFF7840	FLASH_PCROP2AER (3.7.16)	Res.																								PCROP2A_END												
	Factory value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0			
0x1FFF7848	FLASH_WRP2AR (3.7.17)	Res.										WRP2A_END				Res.						WRP2A_STRT																
	Factory value	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1			
0x1FFF7850	FLASH_WRP2BR (3.7.18)	Res.										WRP2B_END				Res.						WRP2B_STRT																
	Factory value	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1			
0x1FFF7858	FLASH_PCROP2BSR (3.7.19)	Res.																								PCROP2B_STRT												
	Factory value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	1	1	1	1	1	1	1	1			



Table 17. Organization of option bytes (continued)

Address <sup>(1)</sup>	Corresponding option register (section)	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
0x1FFF7860	FLASH_PCROP2BER (3.7.20)	Res.																								PCROP2B_END							
	Factory value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0
0x1FFF7870	FLASH_SECR (3.7.21)	Res.				SEC_SIZE2								Res.				BOOT_LOCK	Res.								SEC_SIZE						
	Factory value	X	X	X	X	0	0	0	0	0	0	0	0	0	X	X	X	0	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0

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

### 3.4.2 FLASH option byte programming

After reset, the bits of the *FLASH control register (FLASH\_CR)* related with the option bytes are write-protected as long as its OPTLOCK bit remains high.

To clear OPTLOCK and allow altering the option bytes, proceed as follows:

1. Unlock the FLASH\_CR with the LOCK clearing sequence (refer to *Unlocking the flash memory*)
2. Write OPTKEY1=0x08192A3B of the *FLASH option key register (FLASH\_OPTKEYR)*
3. Write OPTKEY2=0x4C5D6E7F of the *FLASH option key register (FLASH\_OPTKEYR)*

Any wrong sequence sets and locks the option lock bit OPTLOCK until the next system reset. A wrong key sequence causes a bus error and generates a HardFault interrupt.

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

*Note:* Setting the LOCK bit by software automatically sets OPTLOCK, too.

#### Modifying user options

The option bytes are stored in the system memory, which cannot be directly programmed by the user application. They can be accessed through the option registers, the flash memory controller handling the memory erase and program.

To modify the value of user option bytes, proceed as follows:

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.

When the BSY1 bit is cleared, the option bytes in the flash memory are updated, but they are not yet applied to the system. They only become effective when loaded by the option byte loader (OBL).

**Caution:** When the option registers are read, they return the content which was loaded at last OBL. Writing them does not alter their read values. Therefore, avoid using read-modify-write instructions when composing new option byte values to program.



**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 (OBL)

The loading of option bytes into the option registers is performed under system reset upon:

- power-on reset (POR)
- exiting from Standby and Shutdown low-power modes
- setting the OBL\_LAUNCH bit of the *FLASH control register (FLASH\_CR)* by software

The values loaded into the option registers upon reset are immediately effective for configuring the device.

In the flash memory, each word of four option bytes has a complementary word containing inverse information. For a given set of four option bytes, the two words are at adjacent addresses, thus forming a double word.

The option byte loader loads the values of option bytes (flash memory locations) to the option registers. The option registers configure the device. They can be read and written by software. These are:

- FLASH\_OPTR
- FLASH\_PCROP<sub>y</sub>SR (x = A or B, y = 1 or 2)
- FLASH\_PCROP<sub>y</sub>ER (x = A or B, y = 1 or 2)
- FLASH\_WRP<sub>y</sub>R (x = A or B, y = 1 or 2)
- FLASH\_SECR

The option registers also allow modifying the option bytes, see *Modifying user options*.

The option byte loader reads the option bytes by double words. ECC on option words during the option byte load operation is not effective. It is only effective when reading the option bytes by software.

If the option byte loader detects a mismatch between the word and its complement, it does not load the option register with the values of the corresponding option byte. Instead, it loads it with a *mismatch* values and sets the OPTVERR bit. The mismatch values of the option registers disable BOR, set the WRP to *no protection*, the RDP to *level 1*, PCROP to *all memory protected*, and BOOT\_LOCK to *boot from main flash memory*. They are 1 for the remaining option bits.

## 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 programming the RDP option byte and performing an OBL to reload the new RDP option byte. The read protection protects the main flash memory, the option bytes, and the backup registers (TAMP\_BKPxR in TAMP).

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 18](#).

**Table 18. 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 and the backup registers are also accessible by all operations.

#### Level 1: Read protection

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

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

**Caution:** In Level 1 with 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 (bootloader 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.

When the read protection is set during or after (since last power on) the debugger is connected or the MCU boot from the system memory/SRAM, apply a POR (power-on reset) instead of a system reset. Otherwise, any data read triggers a hard fault.

*Note:* If the read protection is programmed through software, do not set the OBL\_LAUNCH bit (FLASH control register (FLASH\_CR) register) to reload the option byte. Instead, perform a POR. This can be done by entering Standby or Shutdown mode then waking up the device.

With the PCROP\_RDP bit of the FLASH PCROP area A end address register (FLASH\_PCROP1AER) set, the change from Level 1 to Level 0 triggers full mass erase of the main flash memory. The backup registers (TAMP\_BKPxR) are also erased. The user options except PCROP protection are set to their previous values copied from FLASH\_OTPR and FLASH\_WRPyR (x = A or B, y = 1 or 2). PCROP is disabled. The OTP area is not affected by mass erase and remains unchanged.

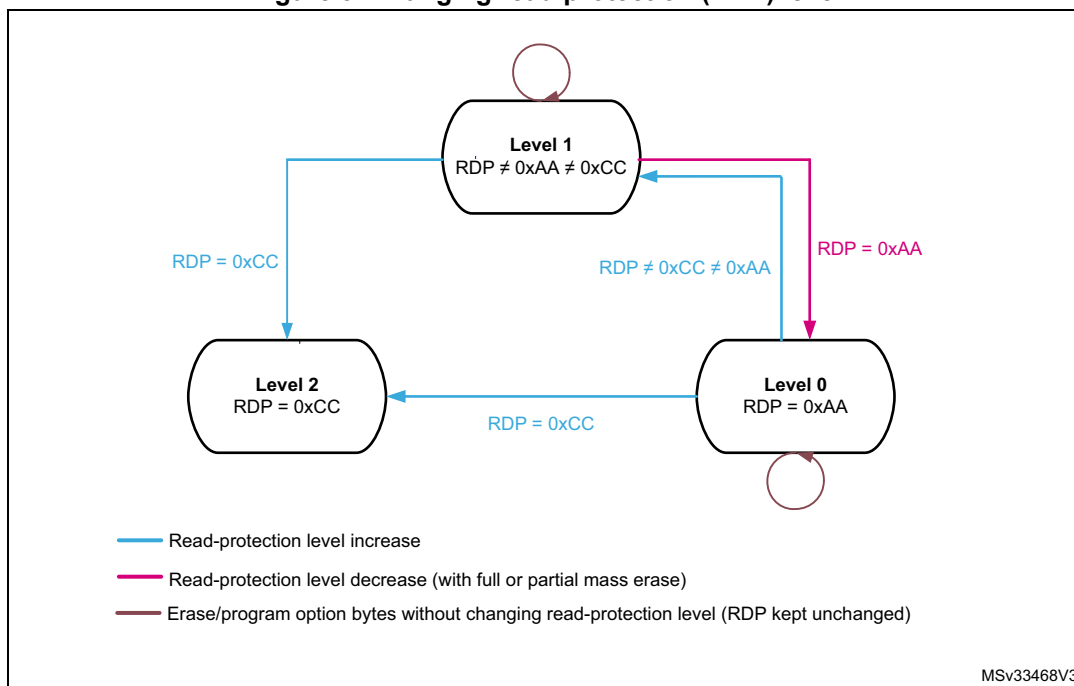
With the PCROP\_RDP bit cleared, a partial mass erase occurs, only erasing flash memory pages that do not overlap with PCROP area (do not contain any PCROP-protected subpage). The option bytes are re-programmed with their previous values. This is also true for FLASH\_PCROP1xSR and FLASH\_PCROP1xER registers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

**Table 19: Mass erase upon RDP regression from Level 1 to Level 0**

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

*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



MSv33468V3

Table 20. 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>
Backup registers	1	Yes	Yes	N/A <sup>(4)</sup>	No	No	N/A <sup>(4)</sup>
	2	Yes	Yes	N/A	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>
OTP	1	Yes	Yes	N/A	Yes	No	N/A
	2	Yes	Yes	N/A	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>	N/A <sup>(1)</sup>

- When the protection Level 2 is active, the Debug port, the boot from RAM and the boot from system memory are disabled.
- The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
- The main flash memory is erased when the RDP option byte is programmed with all level of protections disabled (0xAA).
- The backup registers are cleared when RDP changes from Level 1 to Level 0.

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

Two areas of the flash memory (flash memory bank for dual-bank devices) 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)* for single-bank devices or for bank 1 of dual-bank devices, and in *FLASH PCROP2 area A start address register (FLASH\_PCROP2ASR)*, *FLASH PCROP2 area A end address register (FLASH\_PCROP2AER)*, *FLASH PCROP2 area B start address register (FLASH\_PCROP2BSR)*, and *FLASH PCROP2 area B end address register (FLASH\_PCROP2BER)* for bank 2 of dual-bank devices.

A PCROP area *x* (A or B) of flash memory bank *y* (1 or 2) is defined from the address:

*Flash memory base address + [PCROP<sub>yx</sub>\_STRT x 0x200]* (included)

to the address:

*Flash memory base address + [(PCROP<sub>yx</sub>\_END + 1) x 0x200]* (excluded).

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

$PCROP_{yx\_END} = PCROP_{yx\_STRT} + 1$ .

When

$PCROP_{yx\_END} = PCROP_{yx\_STRT}$ ,

the full flash memory (or bank *y* for dual-bank devices) 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 21: PCROP protection**

PCROP register values (x = A or B, y = 1 or 2)	PCROP-protected area
PCROP <sub>yx</sub> _STRT = PCROP <sub>yx</sub> _END	Full flash memory
PCROP <sub>yx</sub> _STRT > PCROP <sub>yx</sub> _END <sup>(1)</sup>	None (unprotected)
PCROP <sub>yx</sub> _STRT < PCROP <sub>yx</sub> _END	Subpages from PCROP <sub>yx</sub> _STRT to PCROP <sub>yx</sub> _END (read-, write-, and erase-protected); PCROP area boundary pages (erase-protected).

1. The PCROP<sub>yx</sub>\_STRT and PCROP<sub>yx</sub>\_END addresses must not point to the same flash memory page.

*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)*, and, for dual-bank devices, also *FLASH WRP2 area A address register (FLASH\_WRP2AR)* and *FLASH WRP2 area B address register (FLASH\_WRP2BR)*.

The WRP *x* area (x = A, B) of bank *y* (y = 1 or 2) is defined from the address  
Flash memory base address + [WRP<sub>yx</sub>\_STRT x 0x0800] (included)

to the address

Flash memory base address + [(WRP<sub>yx</sub>\_END+1) x 0x0800] (excluded).

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

WRP<sub>yx</sub>\_END = WRP<sub>yx</sub>\_STRT.

For example, to protect the bank 1 of 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 flash memory).

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

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

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

**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 22: WRP protection**

WRP registers values (x = A or B, y= 1 or 2)	WRP-protected area
WRPyx_STRT = WRPyx_END	Page WRPyx
WRPyx_STRT > WRPyx_END	None (unprotected)
WRPyx_STRT < WRPyx_END	Pages from WRPyx_STRT to WRPyx_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 and SEC\_PROT2 bits of the FLASH\_CR register set, the latter pertaining to bank 2 in dual-bank devices), any access to securable memory area (fetch, read, programming, erase) is rejected, generating a bus error. The securable area can only be unsecured by a system reset.

The size of the securable memory area is defined by the SEC\_SIZE[7:0] and SEC\_SIZE2[7:0] (the latter for dual-bank devices only) bitfields of the FLASH\_SECR register. For STM32G071xx and STM32G081xx, it can only be modified in RDP level 0. For all the other devices, it can only be modified in RDP level 0 or 1, with SEC\_PROT = 0 and SEC\_PROT2 = 0 specifically on STM32G0B1xx and STM32G0C1xx. 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 23](#).*



Table 23. Securable memory erase at RDP Level 1 to Level 0 change

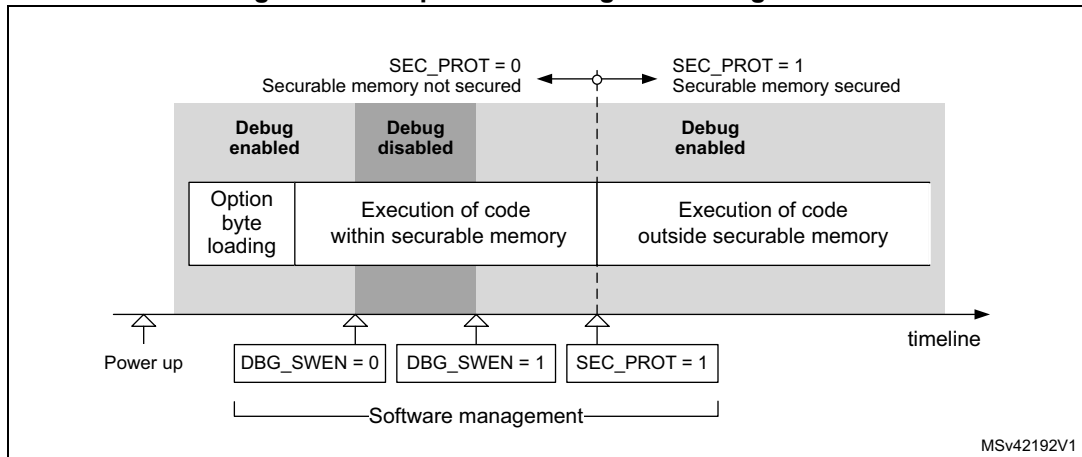
Securable memory size (SEC_SIZEx[7:0])	PCROP_RDP	Erased pages
0	1	All (mass erase)
0	0	All but PCROP
> 0	1	All (mass erase)
> 0	0	All but PCROP outside the securable memory area

### 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 (or SEC\_PROT2, for bank 2 in dual-bank devices) bits.

Figure 4. Example of disabling core debug access



### 3.5.6 Forcing boot from main 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 mode configuration bits or the empty check flag. 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.

**Caution:** If BOOT\_LOCK is set in association with RDP level 1, the debug capabilities of the device are stopped and the reset value of the DBG\_SWEN bit of the FLASH access control register (FLASH\_ACR) becomes zero. If DBG\_SWEN bit is not set by the application code after reset, there is no way to recover from this situation.



### 3.6 FLASH interrupts

Table 24. 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
Data miss during fast programming error	MISSERR	Write MISSERR=1	N/A
Fast programming error	FASTERR	Write FASTERR=1	N/A
ECC error correction	ECCC	Write ECCC=1	ECCCIE
ECC double error (NMI)	ECCD	Write ECCD=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: 0b0000 0000 0000 010X 0000 0110 0000 0000 (EMPTY bit is updated only by OBL, it is not affected by a 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.	DBG_SWEN	Res.	EMPTY
													rw		rw
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.	Res.	Res.	Res.	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 was read as erased or as programmed during OBL. It is not affected by a system reset.

Software may need to change this bit value after a flash memory program or erase operation.

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 configures the number of CPU wait states inserted when accessing the flash memory.  
 000: Zero wait states  
 001: One wait state  
 010: Two wait states  
 Others: 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	BSY2	BSY1
													r	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.

The flag is set and reset by hardware.

The flag high level indicates that a program or erase operation is ongoing, based on the current *FLASH control register (FLASH\_CR)* settings. As long as the flag remains high, the FLASH\_CR register must not be written (this causes HardFault) and the software cannot start another flash memory operation.

**For flash memory program operation**, the flag is set when the first word is sent. It is cleared after the second word is sent, when the operation completes or ends with an error.

**For flash memory erase operation**, the flag is set when setting STRT bit of *FLASH control register (FLASH\_CR)*. It is cleared when the operation completes or ends with an error.

*Note: The flag is also set with the first byte of an access to the locked flash memory. In this case, a complete double word sent to the flash memory must be finished in order to clear the flag.*

Bit 17 **BSY2**: Bank 2 busy

This flag indicates that a flash memory bank 2 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 16 **BSY1**: Bank 1 busy

This flag indicates that a flash memory (bank 1 for dual-bank devices) 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, 32 double words (256 bytes) must be sent to flash memory successively, and the new data must be sent to the logic control before the current data is fully programmed. MISSERR is set by hardware when the new data is not present in time.  
Cleared by writing 1.
- 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

Do not write this register as long as CFGBSY in *FLASH status register (FLASH\_SR)* is high, as such a write causes a HardFault exception.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	SEC_PROT2	SEC_PROT	OBL_LAUNCH	RD_ERRIE	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	FSTPG	OPT_STRT	STRT
rs	rs	rw	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
MER2	Res	BKER	Res	Res	Res	PNB[6:0]						MER1	PER	PG	
rw		rw				rw	rw	rw	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. Furthermore, it is not possible to write the option registers. 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 SEC\_PROT2:** Securable memory area protection enable, bank 2  
 This bit enables the protection on securable area in bank 2, provided that a non-null securable memory area size (SEC\_SIZE2[7:0]) 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 28 SEC\_PROT:** Securable memory area protection enable (bank 1)  
 This bit enables the protection on securable area (in bank 1 for dual-bank devices), provided that a non-null securable memory area size (SEC\_SIZE[7:0]) 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  
 Setting this bit triggers the modification of option bytes.  
 This bit is set only by software, and is cleared when the BSY1 bit is cleared in FLASH\_SR. It cannot be set as long as OPTLOCK is high.
- 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.
- Bit 15 **MER2**: Mass erase, bank 2  
 When set, this bit triggers the mass erase of bank 2 (all user pages).
- Bit 14 Reserved, must be kept at reset value.
- Bit 13 **BKER**: Bank selection for erase operation  
 This bit selects the target of erase operation (bank 1 or bank 2).  
 0: Bank 1  
 1: Bank 2  
 The bit has no effect on the single-bank devices.
- Bits 12:10 Reserved, must be kept at reset value.
- Bits 9:3 **PNB[6:0]**: Page number selection  
 These bits select the page to erase:  
 0x00: page 0  
 0x01: page 1  
 ...  
 0x7F: page 127  
*Note: Values corresponding to addresses outside the main memory are not allowed.*
- Bit 2 **MER1**: Mass erase (bank 1)  
 When set, this bit triggers the mass erase, that is, all user pages (of bank 1 for dual-bank devices).
- 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 ECC register (FLASH\_ECCR)

Address offset: 0x018

Reset value: 0x0000 0000

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

This register applies to single-bank devices and to bank 1 of dual-bank devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	Res.	Res.	SYSF_ECC	Res.	Res.	Res.	Res.
rc_w1	rc_w1						rw				r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ECCD**: ECC detection

Set by hardware when two ECC errors have been detected. When this bit is set, a NMI is generated.

Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

Set by hardware when one ECC error has been detected and corrected. An interrupt is generated if ECCIE is set.

Cleared by writing 1.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **ECCCIE**: ECC correction interrupt enable

0: ECCC interrupt disabled

1: ECCC interrupt enabled

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

Bit 20 **SYSF\_ECC**: System flash memory ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the system flash memory.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR\_ECC[15:0]**: ECC fail double-word address offset

In case of ECC error or ECC correction detected, this bitfield contains double-word offset (multiple of 64 bits) to main flash memory.



### 3.7.7 FLASH ECC register 2 (FLASH\_ECCR2)

Address offset: 0x01C

Reset value: 0x0000 0000

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

This register applies to bank 2 of dual-bank devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	Res.	Res.	SYSF_ECC	Res.	Res.	Res.	Res.
rc_w1	rc_w1						rw				r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ECCD**: ECC detection

Set by hardware when two ECC errors have been detected. When this bit is set, a NMI is generated.

Cleared by writing 1.

Bit 30 **ECCC**: ECC correction

Set by hardware when one ECC error has been detected and corrected. An interrupt is generated if ECCIE is set.

Cleared by writing 1.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **ECCCIE**: ECC correction interrupt enable

0: ECCC interrupt disabled

1: ECCC interrupt enabled

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

Bit 20 **SYSF\_ECC**: System flash memory ECC fail

This bit indicates that the ECC error correction or double ECC error detection is located in the system flash memory.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR\_ECC[15:0]**: ECC fail double-word address offset

In case of ECC error or ECC correction detected, this bitfield contains double-word offset (multiple of 64 bits) to main flash memory.

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

Address offset: 0x020

Reset value: 0b11XX XXXX 1XXX XXXX XXXX XXXX XXXX XXXX

The option bits are loaded with values from flash memory at power-on reset release.

Access: no wait state when no flash memory operation is on going, word, half-word 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]		nBOOT0	nBOOT1	nBOOT_SEL	Res.	RAM PARITY CHECK	DUAL_BANK	nSWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW
		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
nRST_SHDW	nRST_STDBY	nRST_STOP	BORF_LEV[1:0]		BORR_LEV[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 the reset pin
- 1: Internal resets drives the reset pin low until it is seen as low level

Bits 28:27 **NRST\_MODE[1:0]**: PF2-NRST mode

- 00: Reserved
- 01: Reset Input only: a low level on the PF2-NRST pin generates system reset, internal resets not propagated to the PF2-NRST pin
- 10: GPIO: GPIO pad functionality, only internal resets possible
- 11: Bidirectional reset: PF2-NRST pin configured in reset input/output mode (legacy mode)

*Note: For GPIO mode, refer to [Section 7.3.17: Reset pin \(PF2-NRST\) in GPIO mode for additional considerations.](#)*

Bit 26 **nBOOT0**: nBOOT0 option bit

- 0: nBOOT0=0
- 1: nBOOT0=1

Bit 25 **nBOOT1**: Boot configuration

Together with the BOOT0 pin or option bit nBOOT0 (depending on nBOOT\_SEL option bit configuration), this bit selects boot mode from the main flash memory, SRAM or the system memory. Refer to [Section 2.5: Boot configuration](#)

Bit 24 **nBOOT\_SEL**

- 0: BOOT0 signal is defined by BOOT0 pin value (legacy mode)
- 1: BOOT0 signal is defined by nBOOT0 option bit

Bit 23 Reserved, must be kept at reset value.

Bit 22 **RAM\_PARITY\_CHECK**: SRAM parity check control

- 0: SRAM parity check enable
- 1: SRAM parity check disable

- Bit 21 **DUAL\_BANK**: Flash memory bank configuration for 256 Kbyte devices  
This bit is only available on STM32G0B1xx and STM32G0C1xx and only effective for 256-Kbyte devices. Regardless of this bit, the 512-Kbyte devices always operate in dual-bank mode and the 128-Kbyte devices in single-bank mode. Refer to [Table 10](#) and [Table 11](#).  
0: Single-bank flash memory  
1: Dual-bank flash memory  
*Note: This bit has no effect on the erase and write protection (WRP) operations.*
- Bit 20 **nSWAP\_BANK**: Flash memory bank access swap control  
This bit controls the flash memory bank swap. With no swap, the flash memory bank accessed through the address range starting at 0x0800 0000 is the physical bank 1. With swap, it is the bank 2.  
0: Swap (bank 2 mapped from 0x0800 0000)  
1: No swap (bank 1 mapped from 0x0800 0000)  
This bit pertains to dual-bank devices only. In single-bank devices, it is reserved.  
*Note: This bit has no effect on the erase and write protection (WRP) operations.*
- Bit 19 **WWDG\_SW**: Window watchdog selection  
0: Hardware window watchdog  
1: Software window watchdog
- Bit 18 **IWDG\_STDBY**: Independent watchdog counter freeze in Standby mode  
0: Independent watchdog counter is frozen in Standby mode  
1: Independent watchdog counter is running in Standby mode
- Bit 17 **IWDG\_STOP**: Independent watchdog counter freeze in Stop mode  
0: Independent watchdog counter is frozen in Stop mode  
1: Independent watchdog counter is running in Stop mode
- Bit 16 **IWDG\_SW**: Independent watchdog selection  
0: Hardware independent watchdog  
1: Software independent watchdog
- Bit 15 **nRST\_SHDW**  
0: Reset generated when entering the Shutdown mode  
1: No reset generated when entering the Shutdown mode
- Bit 14 **nRST\_STDBY**  
0: Reset generated when entering the Standby mode  
1: No reset generated when entering the Standby mode
- Bit 13 **nRST\_STOP**  
0: Reset generated when entering the Stop mode  
1: No reset generated when entering the Stop mode
- Bits 12:11 **BOR\_LEV[1:0]**: BOR threshold at falling  $V_{DD}$  supply  
Falling  $V_{DD}$  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

- Bits 10:9 **BORR\_LEV[1:0]**: BOR threshold at rising  $V_{DD}$  supply  
 Rising  $V_{DD}$  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\_RISING and BORF\_LEV\_FALLING taken into account
- Bits 7:0 **RDP[7:0]**: Read protection level  
 Note: 0xAA: Level 0, read protection not active  
 0xCC: Level 2, chip read protection active  
 Others: Level 1, memories read protection active

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

Address offset: 0x024

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

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

This register applies to single-bank devices and to bank 1 of dual-bank devices.

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.	PCROP1A_STRT[8:0]										
							rw	rw	rw	rw	rw	rw	rw	rw	rw		

- Bits 31:9 Reserved, must be kept at reset value.
- Bits 8:0 **PCROP1A\_STRT[8:0]**: PCROP1A area start offset (bank 1)  
 Contains the offset of the first subpage of the PCROP1A area (in bank 1 for dual-bank devices).  
 Note: The number of effective bits depends on the size of flash memory in the device.



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

Address offset: 0x028

Reset value: 0bX000 0000 0000 0000 0000 000X XXXX XXXX (The option bits are loaded with values from flash memory at power-on reset release.)

Access: no wait state when no flash memory operation is on going, word, half-word access. 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	PCROP1A_END[8:0]										
							rw	rw	rw	rw	rw	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 30:9 Reserved, must be kept at reset value.

Bits 8:0 **PCROP1A\_END[8:0]**: PCROP1A area end offset (bank 1)

Contains the offset of the last subpage of the PCROP1A area (in bank 1 for dual-bank devices).

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

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

Address offset: 0x02C

Reset value: 0x00XX 00XX (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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_END[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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_STRT[6:0]								
									rw	rw	rw	rw	rw	rw	rw		

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1A\_END[6:0]**: WRP area A end offset (bank 1)

This bitfield contains the offset of the last page of the WRP area A (in bank 1 for dual-bank devices).

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

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

Bits 6:0 **WRP1A\_STRT[6:0]**: WRP area A start offset (bank 1)

This bitfield contains the offset of the first page of the WRP area A (in bank 1 for dual-bank devices).

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

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

Address offset: 0x030

Reset value: 0b0000 0000 0XXX XXXX 0000 0000 0XXX 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 and byte access.

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1B\_END[6:0]**: WRP area B end offset (bank 1)

This bitfield contains the offset of the last page of the WRP area B (in bank 1 for dual-bank devices).

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

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

Bits 6:0 **WRP1B\_STRT[6:0]**: WRP area B start offset (bank 1)

This bitfield contains the offset of the first page of the WRP area B (in bank 1 for dual-bank devices).

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

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

Address offset: 0x034

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

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

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.	PCROP1B_STRT[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:0 **PCROP1B\_STRT[8:0]**: PCROP1B area start offset (bank 1)

Contains the offset of the first subpage of the PCROP1B area (in bank 1 for dual-bank devices)

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

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

Address offset: 0x038

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

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

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.	PCROP1B_END[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:0 **PCROP1B\_END[8:0]**: PCROP1B area end offset (bank 1)

Contains the offset of the last subpage of the PCROP1B area (in bank 1 for dual-bank devices).

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

### 3.7.15 FLASH PCROP2 area A start address register (FLASH\_PCROP2ASR)

Address offset: 0x044

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

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

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.	PCROP2A_STRT[8:0]										
							rw	rw	rw	rw	rw	rw	rw	rw	rw		

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

Bits 8:0 **PCROP2A\_STRT[8:0]**: PCROP2A area start offset, bank 2

Contains the offset of the first subpage of the PCROP2A area in bank 2.

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

### 3.7.16 FLASH PCROP2 area A end address register (FLASH\_PCROP2AER)

Address offset: 0x048

Reset value: 0bX000 0000 0000 0000 0000 000X XXXX XXXX (The option bits are loaded with values from flash memory at power-on reset release.)

Access: no wait state when no flash memory operation is on going, word, half-word access. 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		
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.	PCROP2A_END[8:0]										
							rw	rw	rw	rw	rw	rw	rw	rw	rw		

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

Bits 8:0 **PCROP2A\_END[8:0]**: PCROP2A area end offset, bank 2

Contains the offset of the last subpage of the PCROP2A area in bank 2.

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



### 3.7.17 FLASH WRP2 area A address register (FLASH\_WRP2AR)

Address offset: 0x04C

Reset value: 0b0000 0000 0XXX XXXX 0000 0000 0XXX 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 and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_END[6:0]						
									r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_STRT[6:0]						
									r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP2A\_END[6:0]**: WRP area A end offset, bank 2

This bitfield contains the offset of the last page of the WRP area A in bank 2.

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

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

Bits 6:0 **WRP2A\_STRT[6:0]**: WRP area A start offset, bank 2

This bitfield contains the offset of the first page of the WRP area A in bank 2.

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

### 3.7.18 FLASH WRP2 area B address register (FLASH\_WRP2BR)

Address offset: 0x050

Reset value: 0b0000 0000 0XXX XXXX 0000 0000 0XXX 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 and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_END[6:0]						
									r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_STRT[6:0]						
									r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP2B\_END[6:0]**: WRP area B end offset, bank 2

This bitfield contains the offset of the last page of the WRP area B in bank 2.

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

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

Bits 6:0 **WRP2B\_STRT[6:0]**: WRP area B start offset, bank 2

This bitfield contains the offset of the first page of the WRP area B in bank 2.

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

### 3.7.19 FLASH PCROP2 area B start address register (FLASH\_PCROP2BSR)

Address offset: 0x054

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

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

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.	PCROP2B_STRT[8:0]										
							r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		

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

Bits 8:0 **PCROP2B\_STRT[8:0]**: PCROP2B area start offset, bank 2

Contains the offset of the first subpage of the PCROP2B area in bank 2.

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

### 3.7.20 FLASH PCROP2 area B end address register (FLASH\_PCROP2BER)

Address offset: 0x058

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

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

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.	PCROP2B_END[8:0]										
							r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		

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

Bits 8:0 **PCROP2B\_END[8:0]**: PCROP2B area end offset, bank 2

Contains the offset of the last subpage of the PCROP2B area in bank 2.

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

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

Address offset: 0x080

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Res.	Res.	Res.	Res.	SEC_SIZE2[7:0]								Res.	Res.	Res.	BOOT_LOCK	
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w				r/w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE[7:0]								
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:20 **SEC\_SIZE2[7:0]**: Securable memory area size, bank 2

Contains the number of securable flash memory pages for bank 2 of dual-bank devices.

Read as reserved on single-bank devices.

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

Bits 19:17 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 **BOOT\_LOCK** is set in association with RDP level 1, the debug capabilities of the device are stopped and the reset value of the **DBG\_SWEN** bit of the **FLASH\_ACR** register becomes zero. If **DBG\_SWEN** bit is not set by the application code after reset, there is no way to recover from this situation.

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

Bits 7:0 **SEC\_SIZE[7:0]**: Securable memory area size (bank 1)

Contains the number of securable flash memory pages (for single-bank devices and bank 1 of dual-bank devices).

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

3.7.22 FLASH register map

Table 25. 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	DBG_SWEN	Res	EMPTY	Res	Res	Res	Res	ICRST	Res	ICEN	PRFTEN	Res	Res	Res	Res	Res	Res	LATENCY [2:0]		
	Reset value														1		X					0		1	0						0	0	0	
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		
	Reset value																																	
0x008	FLASH_KEYR	KEYR[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00C	FLASH_OPT_KEYR	OPTKEY[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x010	FLASH_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CFGBSY	BSY2	BSY1	OPTVERR	RDERR	Res	Res	Res	Res	Res	FASTERR	MISERR	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res	OPERR	EOP
	Reset value														0	0	0	X	0					0	0	0	0	0	0	0	0	0	0	0
0x014	FLASH_CR	LOCK	OPTLOCK	SEC_PROT2	SEC_PROT	OBL_LAUNCH	RDERRIE	ERRIE	EOPIE	Res	Res	Res	Res	Res	FSTPG	OPTSTRT	STRT	MER2	Res	BKER	Res	Res	Res	PNB[6:0]						MER1	PER	PG		
	Reset value	1	1	0	0	0	0	0	0						0	0	0	0		0				0	0	0	0	0	0	0	0	0	0	0
0x018	FLASH_ECCR	ECCD	ECCC	Res	Res	Res	Res	Res	ECCDIE	Res	Res	Res	Res	Res	Res	Res	Res	Res	ADDR_ECC[15:0]															
	Reset value	0	0						0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x01C	FLASH_ECCR2	ECCD	ECCC	Res	Res	Res	Res	Res	ECCDIE	Res	Res	Res	Res	Res	Res	Res	Res	Res	ADDR_ECC[15:0]															
	Reset value	0	0						0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x020	FLASH_OPTR	Res	Res	IRHEN	NRST_MODE[1:0]	nBOOT0	nBOOT1	nBOOT_SEL	Res	RAM_PARITY_CHECK	DUAL_BANK	nSWAP_BANK	WWDG_SW	IWDG_STBY	IWDG_STOP	IWDG_SW	nRST_SHDW	nRST_STDBY	nRST_STOP	BORF_LEV[1:0]	BORR_LEV[1:0]	BORR_LEV[1:0]	BOR_EN	RDP[7:0]										
	Reset value			X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x024	FLASH_PCROP1ASR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PCROP1A_STRT[8:0]								
	Reset value																							X	X	X	X	X	X	X	X	X	X	
0x028	FLASH_PCROP1AER	PCROP_RDP	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PCROP1A_END[8:0]								
	Reset value	X																							X	X	X	X	X	X	X	X	X	
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	WRP1A_END[6:0]							
	Reset value																										X	X	X	X	X	X	X	X

Table 25. FLASH register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x030	FLASH_WRP1BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_END[6:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_START[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X	X
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.	PCROP1B_START[8:0]							
	Reset value																									X	X	X	X	X	X	X	X	X
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.	PCROP1B_END[8:0]							
	Reset value																									X	X	X	X	X	X	X	X	X
0x03C - 0x043	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.	Res.	
0x044	FLASH_PCROP2ASR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2A_START[8:0]							
	Reset value																										X	X	X	X	X	X	X	X
0x048	FLASH_PCROP2AER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2A_END[8:0]							
	Reset value																									X	X	X	X	X	X	X	X	X
0x04C	FLASH_WRP2AR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_END[6:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_START[6:0]					
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X	X
0x050	FLASH_WRP2BR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_END[6:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_START[6:0]					
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X	X
0x054	FLASH_PCROP2BSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2B_START[8:0]							
	Reset value																									X	X	X	X	X	X	X	X	X
0x058	FLASH_PCROP2BER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCROP2B_END[8:0]							
	Reset value																									X	X	X	X	X	X	X	X	X
0x05C - 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.	Res.	
0x080	FLASH_SECR	Res.	Res.	Res.	Res.	SEC_SIZE2[7:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_SIZE[7:0]							
	Reset value					X	X	X	X	X	X	X	X					X									X	X	X	X	X	X	X	X

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

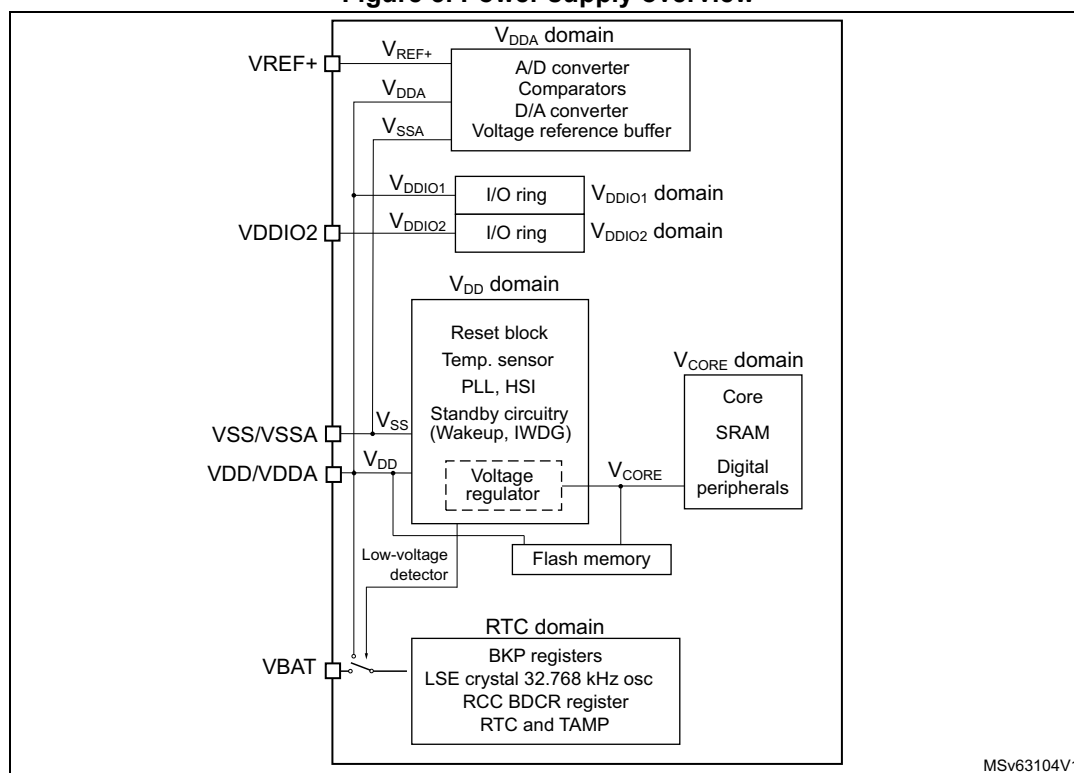
## 4 Power control (PWR)

### 4.1 Power supplies

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

- $V_{DD} = 1.7 \text{ V (1.60 V) to } 3.6 \text{ V}$   
 $V_{DD}$  is the external power supply for the internal regulator and the system analog such as reset, power management and internal clocks. It is provided externally through VDD/VDDA pin.  
 Note that the minimum voltage of 1.7 V corresponds to power-on reset release threshold  $V_{POR(MAX)}$ . Once this threshold is crossed and power-on reset is released, the functionality is guaranteed down to power-down reset threshold  $V_{PDR(MIN)}$ .
- $V_{DDA} = 1.62 \text{ V (ADC and COMP) / } 1.8 \text{ V (DAC) / } 2.4 \text{ V (VREFBUF) to } 3.6 \text{ V}$   
 $V_{DDA}$  is the analog power supply for the A/D converter, D/A converters, voltage reference buffer and comparators.  $V_{DDA}$  voltage level is identical to  $V_{DD}$  voltage as it is provided externally through VDD/VDDA pin.
- $V_{DDIO1} = V_{DD}$   
 $V_{DDIO1}$  is the power supply for the I/Os.  $V_{DDIO1}$  voltage level is identical to  $V_{DD}$  voltage as it is provided externally through VDD/VDDA pin.
- $V_{DDIO2} = 1.6 \text{ to } 3.6 \text{ V (only available on STM32G0B1xx and STM32G0C1xx)}$   
 $V_{DDIO2}$  is the power supply from VDDIO2 pin for selected I/Os. Although  $V_{DDIO2}$  is independent of  $V_{DD}$  or  $V_{DDA}$ , it must not be applied without valid  $V_{DD}$ .
- $V_{BAT} = 1.55 \text{ V to } 3.6 \text{ V}$   
 $V_{BAT}$  is the power supply (through a power switch) for RTC, TAMP, low-speed external 32.768 kHz oscillator and backup registers when  $V_{DD}$  is not present.  $V_{BAT}$  is provided externally through VBAT pin. When this pin is not available on the package, it is internally bonded to VDD/VDDA.
- $V_{REF+}$  is the input reference voltage for the ADC and DAC, or the output of the internal voltage reference buffer (when enabled). When  $V_{DDA} < 2 \text{ V}$ ,  $V_{REF+}$  must be equal to  $V_{DDA}$ . When  $V_{DDA} \geq 2 \text{ V}$ ,  $V_{REF+}$  must be between 2 V and  $V_{DDA}$ . It can be grounded when the ADC and DAC are not active.  
 The internal voltage reference buffer supports two output voltages, which is configured with VRS bit of the VREFBUF\_CSR register:
  - $V_{REF+}$  around 2.048 V (requiring  $V_{DDA}$  equal to or higher than 2.4 V)
  - $V_{REF+}$  around 2.5 V (requiring  $V_{DDA}$  equal to or higher than 2.8 V) $V_{REF+}$  is delivered through VREF+ pin. On packages without VREF+ pin,  $V_{REF+}$  is internally connected with  $V_{DD}$ , and the internal voltage reference buffer must be kept disabled (refer to datasheets for package pinout description).
- $V_{CORE}$  is an internal supply for digital peripherals, SRAM and flash memory. It is produced by an embedded linear voltage regulator. On top of  $V_{CORE}$ , the flash memory is also powered from  $V_{DD}$ .

Figure 5. Power supply overview



MSv63104V1

#### 4.1.1 ADC and DAC reference voltage

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

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

The internal buffered voltage reference is enabled by setting the ENVR bit in the [VREFBUF control and status register \(VREFBUF\\_CSR\)](#). The internal buffered voltage reference is set to 2.5 V when the VRS bit is set and to 2.048 V when the VRS bit is cleared. The internal buffered voltage reference can also provide the voltage to external components through  $V_{REF+}$  pin. Refer to the device datasheet and to [Section 17: Voltage reference buffer \(VREFBUF\)](#) for further information.

#### 4.1.2 Battery backup of RTC domain

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

The VBAT pin powers the RTC and TAMP units, the LSE oscillator and the PC13 to PC15 I/Os, allowing the RTC and TAMP to operate even when the main power supply is turned off. The switch to the  $V_{BAT}$  supply is controlled by the power-down reset embedded in the Reset block.

---

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

---

If no external battery is used in the user application, it is recommended to connect VBAT pin externally to VDD/VDDA pin with a 100 nF external ceramic decoupling capacitor.

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

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

- PC13, PC14 and PC15 can be controlled only by RTC, TAMP or LSE (refer to [Section 30.3: RTC functional description](#))
- RTC\_OUT1 function on PC13
- RTC\_TS function on PC13 or PA4
- TAMP\_IN1 function on PC13 or PA4 and TAMP\_IN2 function on PA0

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

### RTC domain access

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

1. Enable the power interface clock by setting the PWREN bits of the [APB peripheral clock enable register 1 \(RCC\\_APBENR1\)](#).
2. Set the DBP bit of the [Power control register 1 \(PWR\\_CR1\)](#) to enable access to the RTC domain.
3. Select the RTC clock source in the [RTC domain control register \(RCC\\_BDCR\)](#).
4. Enable the RTC clock by setting the RTCEN bit in the [RTC domain control register \(RCC\\_BDCR\)](#).



### VBAT battery charging

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

The VBAT charging is done either through a 5 k $\Omega$  resistor or through a 1.55 k $\Omega$  resistor depending on the VBRS bit value in the PWR\_CR4 register.

The battery charging is enabled by setting VBE bit in the PWR\_CR4 register. It is automatically disabled in VBAT mode.

### 4.1.3 Voltage regulator

Two embedded linear voltage regulators supply all the digital circuitries, except for the Standby circuitry and the RTC domain. The main regulator output voltage ( $V_{CORE}$ ) can be programmed by software to two different power ranges (Range 1 and Range 2) in order to optimize the consumption depending on the system maximum operating frequency (refer to [Section 5.2.8: Clock source frequency versus voltage scaling](#) and to [Section 3.3.4: FLASH read access latency](#)).

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

- In Run, Sleep and Stop 0 modes, both regulators are enabled and the main regulator (MR) supplies full power to the  $V_{CORE}$  domain (core, memories and digital peripherals).
- In Low-power run and Low-power sleep modes, the main regulator is off and the low-power regulator (LPR) supplies low-power to the  $V_{CORE}$  domain, preserving the contents of the registers and SRAM.
- In Stop 1 mode, the main regulator is off and the low-power regulator (LPR) supplies low-power to the  $V_{CORE}$  domain, preserving the contents of the registers and SRAM.
- In Standby mode with SRAM content preserved (RRS bit is set in the PWR\_CR3 register), the main regulator (MR) is off and the low-power regulator (LPR) provides the supply to SRAM only. The core and digital peripherals (except Standby circuitry and RTC domain) are powered off.
- In Standby mode, both regulators are powered off. The contents of the registers and SRAM is lost except for the Standby circuitry and the RTC domain.
- In Shutdown mode, both regulators are powered off. When exiting Shutdown mode, a power-on reset is generated. Consequently, the contents of the registers and SRAM is lost, except for the RTC domain.

#### 4.1.4 Dynamic voltage scaling management

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

Dynamic voltage scaling to increase  $V_{CORE}$  is known as overvolting. It allows to improve the device performance.

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

Two voltage ranges are available:

- Range 1: High-performance range

The main regulator provides a typical output voltage at 1.2 V. The system clock frequency can be up to 64 MHz. The flash memory access time for read access is minimum, write and erase operations are possible.

- Range 2: Low-power range

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

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

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

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

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

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

## 4.2 Power supply supervisor

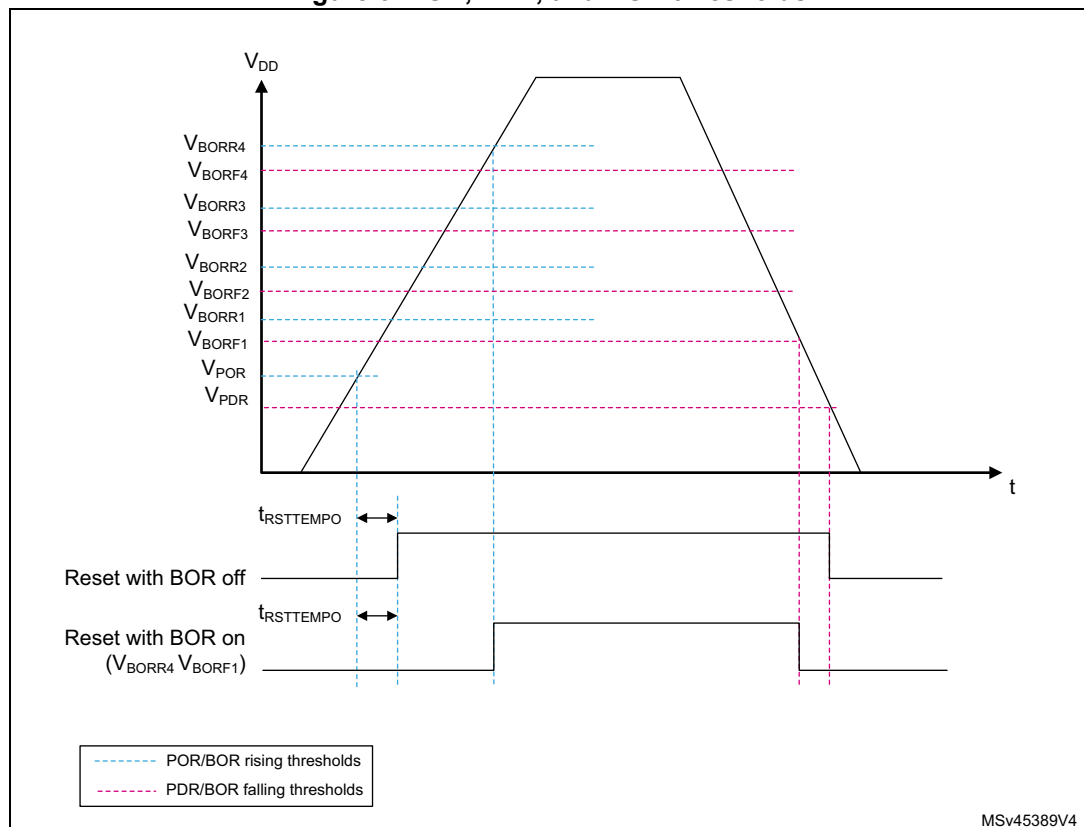
### 4.2.1 Power-on reset (POR) / power-down reset (PDR) / brown-out reset (BOR)

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

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

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

Figure 6. POR, PDR, and BOR thresholds



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

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

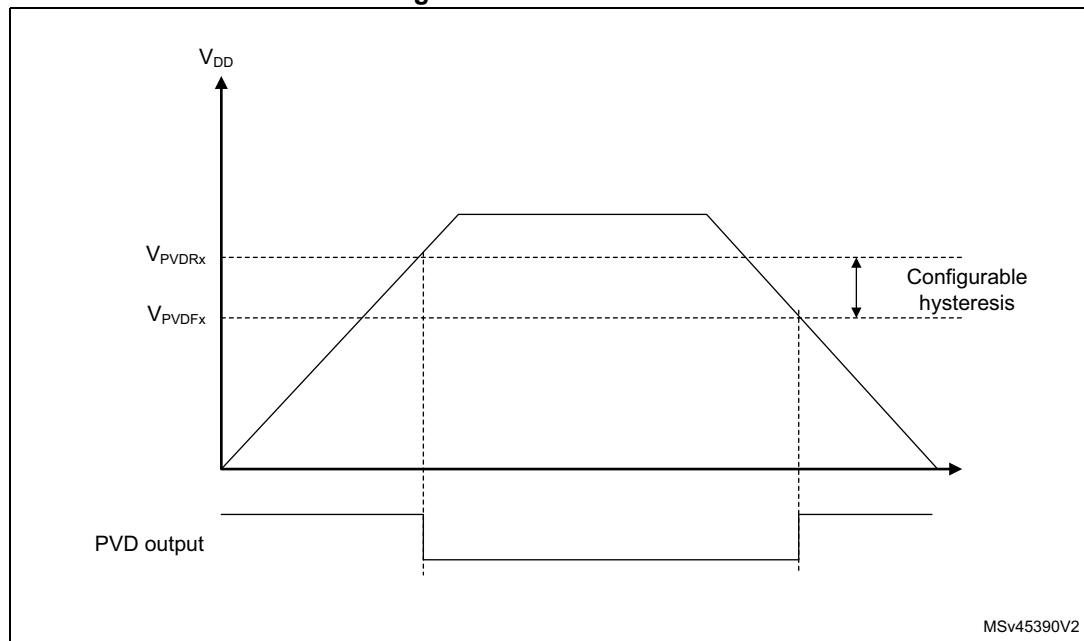
### 4.2.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the  $V_{DD}$  power supply by comparing it to the thresholds selected through PVDRT[2:0] bits (rising thresholds) and PVDFT[2:0] bits (falling thresholds) in the *Power control register 2 (PWR\_CR2)*.  $V_{PVDFx}$  should always be set to a lower voltage level than  $V_{PVDRx}$ .

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available in the *Power status register 2 (PWR\_SR2)*. It indicates if  $V_{DD}$  is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when  $V_{DD}$  drops below the PVD threshold and/or when  $V_{DD}$  rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example, the service routine could perform emergency shutdown tasks.

Figure 7. PVD thresholds



## 4.3 Low-power modes

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

The device features seven low-power modes:

- Sleep mode: CPU clock off, all peripherals including Cortex<sup>®</sup>-M0+ core peripherals such as NVIC, SysTick, etc. can run and wake up the CPU when an interrupt or an event occurs. Refer to [Section 4.3.4: Sleep mode](#).
- Low-power run mode: This mode is achieved when the system clock frequency is reduced below 2 MHz. The code is executed from the SRAM or the flash memory. The regulator is in low-power mode to minimize the regulator's operating current. Refer to [Section 4.3.2: Low-power run mode \(LP run\)](#).
- Low-power sleep mode: This mode is entered from the Low-power run mode: Cortex<sup>®</sup>-M0+ is off. Refer to [Section 4.3.5: Low-power sleep mode \(LP sleep\)](#).
- Stop 0 and Stop 1 modes: SRAM and all registers content are retained. All clocks in the  $V_{CORE}$  domain are stopped, the PLL, the HSI16 and the HSE are disabled. The LSI and the LSE can be kept running.

The RTC and TAMP can remain active (Stop mode with RTC, Stop mode without RTC). Some peripherals with the wake-up capability can enable the HSI16 RC during the Stop mode to detect their wake-up condition.

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

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

Refer to [Section 4.3.6: Stop 0 mode](#) for details on Stop 0 mode.

- Standby mode:  $V_{CORE}$  domain is powered off.  
However, it is possible to preserve SRAM content:
  - Standby mode with SRAM retention when the bit RRS is set in PWR\_CR3 register. In this case, SRAM is supplied by the low-power regulator.
  - Standby mode when the bit RRS is cleared in PWR\_CR3 register. In this case the main regulator and the low-power regulator are powered off.

All clocks in the  $V_{CORE}$  domain are stopped and the PLL, the HSI16, and the HSE oscillators are disabled. The LSI and the LSE oscillators can be kept running.

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

The system clock, when exiting Standby mode, is the HSI16 oscillator clock.

Refer to [Section 4.3.8: Standby mode](#).

- Shutdown mode:  $V_{CORE}$  domain is powered off. All clocks in the  $V_{CORE}$  domain are stopped, the PLL, the HSI16, the LSI and the HSE oscillators are disabled. The LSE can be kept running. The system clock, when exiting Shutdown mode, is the HSI16 oscillator clock. In this mode, the supply voltage monitoring is disabled and the product

behavior is not guaranteed in case of a power voltage drop. Refer to [Section 4.3.9: Shutdown mode](#).

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

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

### Peripheral clock gating

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

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

The peripheral clock gating is controlled by the RCC\_AHBENR and RCC\_APBENRx registers.

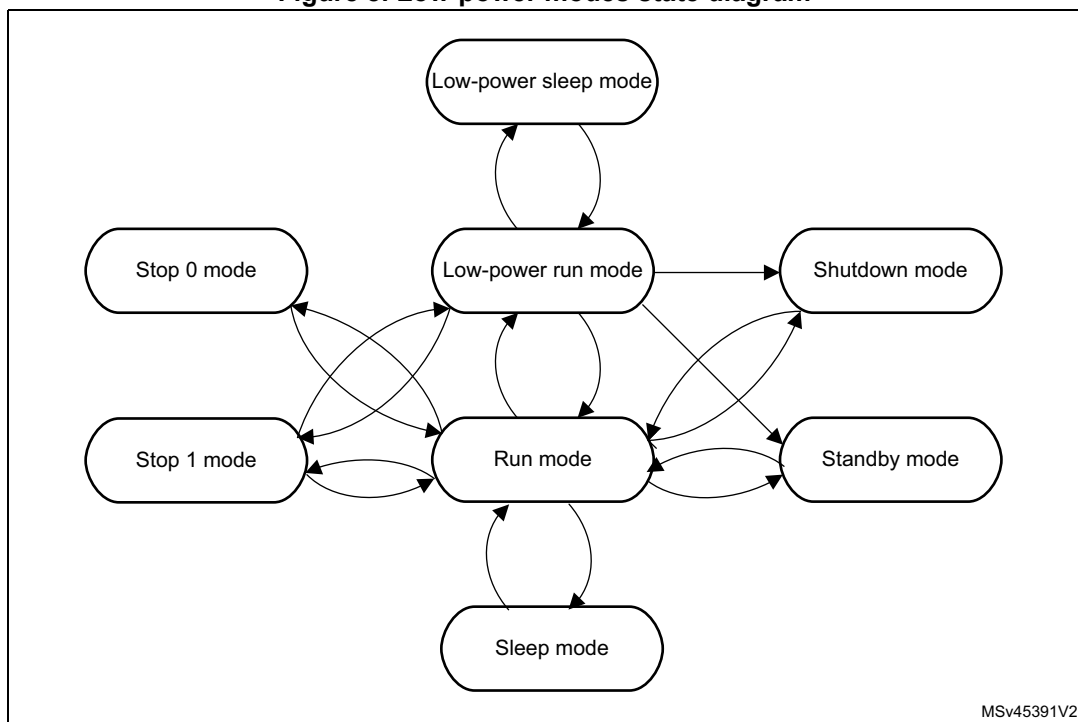
Disabling the peripherals clocks in Sleep/Stop modes can be performed automatically by resetting the corresponding bit in the RCC\_AHBSMENR and RCC\_APBSMENRx registers.

### Debug mode

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

However, by setting some configuration bits in the DBG\_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to [Section 40.9.1: Debug support for low-power modes](#).

Figure 8. Low-power modes state diagram



MSv45391V2

**Table 26. Low-power mode summary**

Mode name	Entry	Wake-up source <sup>(1)</sup>	Wake-up system clock	Effect on clocks	Voltage regulators	
					MR	LPR
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	CPU clock OFF no effect on other clocks or analog clock sources	ON	ON
	WFE	Wake-up event				
Low-power run	Set LPR bit	Clear LPR bit	No change	None	OFF	
Low-power sleep	Set LPR bit + WFI or Return from ISR	Any interrupt	Same as before entering Low-power sleep mode	CPU clock OFF no effect on other clocks or analog clock sources		
	Set LPR bit + WFE	Wake-up event				
Stop 0	LPMS="000" + SLEEPDEEP bit + WFI or Return from ISR or WFE	Any EXTI line (configured in the EXTI registers)  Specific peripherals events	HSISYS	All clocks OFF except LSI and LSE	ON	
Stop 1	LPMS="001" + SLEEPDEEP bit + WFI or Return from ISR or WFE				OFF	
Standby with SRAM	LPMS="011" + Set RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, TAMP event, external reset on NRST pin, IWDG reset			OFF	
Standby	LPMS="011" + Clear RRS bit + SLEEPDEEP bit + WFI or Return from ISR or WFE					
Shutdown	LPMS="1--" + SLEEPDEEP bit + WFI or Return from ISR or WFE				OFF	

1. Refer to [Table 27: Functionalities depending on the working mode.](#)

Table 27. Functionalities depending on the working mode<sup>(1)</sup>

Function	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Standby		Shutdown		VBAT
					-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
CPU	Y	-	Y	-	-	-	-	-	-	-	-
Flash memory	Y	Y	O <sup>(2)</sup>	O <sup>(2)</sup>	O <sup>(2)</sup>	-	-	-	-	-	-
SRAM	Y	Y <sup>(3)</sup>	Y	Y <sup>(3)</sup>	Y	-	O <sup>(4)</sup>	-	-	-	-
Backup Registers	Y	Y	Y	Y	Y	-	Y	-	Y	-	Y
BOR	Y	Y	Y	Y	Y	Y	Y	Y	-	-	-
PVD	O	O	O	O	O	O	-	-	-	-	-
DMA1/2	O	O	O	O	-	-	-	-	-	-	-
HSI16	O	O	O	O	<sup>(5)</sup>	-	-	-	-	-	-
HSI48	O	O	-	-	-	-	-	-	-	-	-
HSE	O	O	O	O	-	-	-	-	-	-	-
LSI	O	O	O	O	O	-	O	-	-	-	-
LSE	O	O	O	O	O	-	O	-	O	-	O
PLL	O	O	-	-	-	-	-	-	-	-	-
CSS	O	O	O <sup>(6)</sup>	O <sup>(6)</sup>	-	-	-	-	-	-	-
CSS on LSE	O	O	O	O	O	O	O	O	-	-	-
RTC / Auto wake-up	O	O	O	O	O	O	O	O	O	O	O
TAMP1/2/3	O	O	O	O	O	O	O	O	O	O	O
USART (FULL <sup>(7)</sup> )	O	O	O	O	O <sup>(8)</sup>	O <sup>(8)</sup>	-	-	-	-	-
USART (BASIC <sup>(7)</sup> )	O	O	O	O	-	-	-	-	-	-	-
LPUART1/2	O	O	O	O	O <sup>(8)</sup>	O <sup>(8)</sup>	-	-	-	-	-
I2C1	O	O	O	O	O <sup>(9)</sup>	O <sup>(9)</sup>	-	-	-	-	-
I2C2	O	O	O	O	O <sup>(9)(10)</sup>	O <sup>(9)(10)</sup>	-	-	-	-	-
I2C3	O	O	O	O	-	-	-	-	-	-	-
SPI1/2/3	O	O	O	O	-	-	-	-	-	-	-
ADC	O	O	O	O	-	-	-	-	-	-	-
DAC	O	O	O	O	O	-	-	-	-	-	-
VREFBUF	O	O	O	O	O	-	-	-	-	-	-



Table 27. Functionalities depending on the working mode<sup>(1)</sup> (continued)

Function	Run	Sleep	Low-power run	Low-power sleep	Stop 0/1		Standby		Shutdown		VBAT
					-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
COMP1/2/3	O	O	O	O	O	O	-	-	-	-	-
Temperature sensor	O	O	O	O	-	-	-	-	-	-	-
TIMx	O	O	O	O	-	-	-	-	-	-	-
LPTIM1/2	O	O	O	O	O	O	-	-	-	-	-
IWDG	O	O	O	O	O	O	O	O	-	-	-
WWDG	O	O	O	O	-	-	-	-	-	-	-
SysTick timer	O	O	O	O	-	-	-	-	-	-	-
RNG	O	O	O	O	-	-	-	-	-	-	-
AES	O	O	O	O	-	-	-	-	-	-	-
CRC	O	O	O	O	-	-	-	-	-	-	-
USB	O	O	-	-	-	O <sup>(11)</sup>	-	-	-	-	-
FDCAN1/2	O	O	-	-	-	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	<sup>(12)</sup>	up to 5 pins <sup>(13)</sup>	<sup>(14)</sup>	up to 5 pins <sup>(13)</sup>	-

- Legend: Y = Yes (Enable). O = Optional (Disable by default. Can be enabled by software). - = Not available.
- The flash memory can be configured in power-down mode. By default, it is not in power-down mode.
- The SRAM clock can be gated ON or OFF.
- SRAM content is preserved when the bit RRS is set in PWR\_CR3 register.
- Some peripherals with wake-up from Stop capability can request HSI16 to be enabled. In this case, HSI16 is woken up by the peripheral, and only feeds the peripheral which requested it. HSI16 is automatically put OFF when the peripheral does not need it anymore.
- If CSS is used on HSE clock in Low power run or Low power sleep modes, configure HSIDIV such as not to drive SYSCLK clock above the maximum frequency for either mode, in case of external clock failure detection.
- Refer to [Table 183: Instance implementation on STM32G0x1](#) for USART instances supporting FULL and BASIC feature set.
- USART and LPUART reception is functional in Stop mode, and generates a wake-up interrupt on Start, address match or received frame event.
- I2C address detection is functional in Stop mode, and generates a wake-up interrupt in case of address match.
- Only for STM32G0B1xx and STM32G0C1xx. Refer to [Table 165: I2C implementation](#) for more detail.
- USB bus state monitoring is functional in Stop mode. It generates a wake-up interrupt upon resume from USB suspend.
- I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
- I/Os with wake-up from Standby/Shutdown mode capability (WKUPx).



14. I/Os can be configured with internal pull-up, pull-down or floating in Shutdown mode but the configuration is lost when exiting Shutdown mode.

### 4.3.1 Run mode

#### Slowing down system clocks

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

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

### 4.3.2 Low-power run mode (LP run)

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

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

#### I/O states in Low-power run mode

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

#### Entering Low-power run mode

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

1. Optional: Jump into the SRAM and power-down the flash memory by setting the FPD\_LPRUN bit in the [Power control register 1 \(PWR\\_CR1\)](#).
2. Decrease the system clock frequency below 2 MHz.
3. Force the regulator in low-power mode by setting the LPR bit in the PWR\_CR1 register.

Refer to [Table 28: Low-power run](#) on how to enter Low-power run mode.

#### Exiting Low-power run mode

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

1. Force the regulator in main mode by clearing the LPR bit in the [Power control register 1 \(PWR\\_CR1\)](#).
2. Wait until REGLPF bit is cleared in the [Power status register 2 \(PWR\\_SR2\)](#).
3. Increase the system clock frequency.

Refer to [Table 28: Low-power run](#) on how to exit Low-power run mode.

Table 28. Low-power run

Low-power run mode	Description
Mode entry	Decrease the system clock frequency below 2 MHz LPR = 1
Mode exit	LPR = 0 Wait until REGLPF = 0 Increase the system clock frequency
Wake-up latency	Regulator wake-up time from low-power mode

### 4.3.3 Low-power modes

#### Entering low-power modes

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

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

#### Exiting low-power modes

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

- If the WFI instruction or Return from ISR was used to enter low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter low-power mode, the device exits low-power mode as soon as an event occurs. The wake-up event can be generated either by:
  - NVIC IRQ interrupt.
 

When SEVONPEND = 0 in the Cortex<sup>®</sup>-M0+ system control register: by enabling an interrupt in the peripheral control register and in the NVIC. When the device resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

When SEVONPEND = 1 in the Cortex<sup>®</sup>-M0+ system control register: by enabling an interrupt in the peripheral control register and optionally in the NVIC. When the device resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

All NVIC interrupts wake the device up, even the disabled ones.
  - Event
 

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

The device exits Standby and Shutdown low-power modes upon an external reset (NRST pin), a rising or falling edge on one of enabled WKUPx pins, or upon an RTC or TAMP event. On top of these, it exits Standby mode also upon an IWDG reset. See [Figure 282](#):

[RTC block diagram](#).

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

### 4.3.4 Sleep mode

#### I/O states in Sleep mode

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

#### Entering Sleep mode

The device enters Sleep mode according to section [Entering low-power modes](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M0+ System Control register is clear.

Refer to [Table 29: Sleep mode summary](#) for details on how to enter Sleep mode.

#### Exiting Sleep mode

The device exits Sleep mode according to [Exiting low-power modes](#).

Refer to [Table 29: Sleep mode summary](#) for more details on how to exit Sleep mode.

**Table 29. Sleep mode summary**

Characteristic	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP = 0</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M0+ system control register.
	On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP = 0 and</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M0+ system control register.
Mode exit	If WFI or return from ISR was used for entry Interrupt: refer to <a href="#">Table 61: Vector table</a> If WFE was used for entry and SEVONPEND = 0: Wake-up event: refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a> If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to <a href="#">Table 61: Vector table</a> or Wake-up event: refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a>
Wake-up latency	None

### 4.3.5 Low-power sleep mode (LP sleep)

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

#### I/O states in Low-power sleep mode

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

### Entering Low-power sleep mode

The device enters Low-power sleep mode from Low-power run mode according to [Entering low-power modes](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M0+ System Control register is clear.

Refer to [Table 30: Low-power sleep mode summary](#) for details on how to enter Low-power sleep mode.

### Exiting Low-power sleep mode

The device exits Low-power sleep mode according to [Exiting low-power modes](#). When exiting Low-power sleep mode by issuing an interrupt or an event, the device is in Low-power run mode.

Refer to [Table 30: Low-power sleep mode summary](#) for details on how to exit Low-power sleep mode.

**Table 30. Low-power sleep mode summary**

Characteristic	Description
Mode entry	Low-power sleep mode is entered from the Low-power run mode. WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP = 0</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M0+ System Control register.
	Low-power sleep mode is entered from the Low-power run mode. On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP = 0 and</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> </ul> Refer to the Cortex <sup>®</sup> -M0+ System Control register.
Mode exit	If WFI or Return from ISR was used for entry Interrupt: refer to <a href="#">Table 61: Vector table</a> If WFE was used for entry and SEVONPEND = 0: Wake-up event: refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a> If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC: refer to <a href="#">Table 61: Vector table</a> Wake-up event: refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a> After exiting Low-power sleep mode, the device is in Low-power run mode.
Wake-up latency	None

### 4.3.6 Stop 0 mode

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

SRAM and register contents are preserved.

The BOR is available in Stop 0 mode.

The BOR and PDR can be activated to sample periodically the supply voltage. This option enabled by setting the ENB\_ULP bit of the PWR\_CR3 register allows decreasing the current consumption in this mode, but any drop of the voltage below the operating conditions between two active periods of the supply detector results in a non-generation of PDR reset.

### I/O states in Stop 0 mode

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

### Entering Stop 0 mode

The device enters Stop 0 mode according to section [Entering low-power modes](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M0+ System Control register is set.

Refer to [Table 31: Stop 0 mode summary](#) for details on how to enter Stop 0 mode.

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

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

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except upon a reset. See [Section 28.3: IWDG functional description](#).
- real-time clock (RTC): this is configured by the RTCEN bit in the [RTC domain control register \(RCC\\_BDCR\)](#).
- Internal RC oscillator (LSI): this is configured by the LSION bit in the [Control/status register \(RCC\\_CSR\)](#).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the [RTC domain control register \(RCC\\_BDCR\)](#).

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

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

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

### Exiting Stop 0 mode

The device exits Stop 0 mode according to section [Entering low-power modes](#).

Refer to [Table 31: Stop 0 mode summary](#) for details on how to exit Stop 0 mode.

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

When exiting Stop 0 mode, the device is either in Run mode (Range 1 or Range 2 depending on VOS bit in PWR\_CR1) or in Low-power run mode if the bit LPR is set in the [Power control register 1 \(PWR\\_CR1\)](#).

**Table 31. Stop 0 mode summary**

Characteristic	Description
Mode entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “000” in PWR_CR1</li> </ul>
	<p>On Return from ISR while:</p> <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ System Control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “000” in PWR_CR1</li> </ul>
	<p><i>Note: To enter Stop 0 mode, all EXTI Line pending bits (in <a href="#">EXTI rising edge pending register 1 (EXTI_RPR1)</a> and <a href="#">EXTI falling edge pending register 1 (EXTI_FPR1)</a>), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and program execution continues.</i></p>
Mode exit	<p>If WFI or Return from ISR was used for entry Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to <a href="#">Table 61: Vector table</a>.</p> <p>If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a>.</p> <p>If WFE was used for entry and SEVONPEND = 1: Any EXTI Line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to <a href="#">Table 61: Vector table</a>.</p> <p>Wake-up event: refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a></p>
Wake-up latency	<p>Longest wake-up time between HSI16 wake-up time and flash memory wake-up time from Stop 0 mode.</p>

### 4.3.7 Stop 1 mode

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

Refer to [Table 32: Stop 1 mode summary](#) for details on how to enter and exit Stop 1 mode.

**Table 32. Stop 1 mode summary**

Characteristic	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “001” in PWR_CR1</li> </ul>
	On Return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ System Control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “001” in PWR_CR1</li> </ul>
	<i>Note: To enter Stop 1 mode, all EXTI Line pending bits (in <a href="#">EXTI rising edge pending register 1 (EXTI_RPR1)</a> and <a href="#">EXTI falling edge pending register 1 (EXTI_FPR1)</a>), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 1 mode entry procedure is ignored and program execution continues.</i>
Mode exit	If WFI or Return from ISR was used for entry Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to <a href="#">Table 61: Vector table</a> . If WFE was used for entry and SEVONPEND = 0: Any EXTI Line configured in event mode. Refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a> . If WFE was used for entry and SEVONPEND = 1: Any EXTI Line configured in Interrupt mode (even if the corresponding EXTI Interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability. Refer to <a href="#">Table 61: Vector table</a> . Wake-up event: refer to <a href="#">Section 13.3.2: EXTI direct event input wake-up</a>
Wake-up latency	Longest wake-up time between HSI16 wake-up time and regulator wake-up time from Low-power mode + flash memory wake-up time from Stop 1 mode.



### 4.3.8 Standby mode

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

The content of the registers is lost except for the registers in the RTC domain and Standby circuitry (see [Figure 5](#)). The SRAM content is lost except if the RRS bit is set in the PWR\_CR3 register. In this case the low-power regulator is on and provides the supply to the SRAM only.

The BOR is available in Standby mode.

The BOR and PDR can be activated to sample periodically the supply voltage. This option enabled by setting the ENB\_ULP bit of the PWR\_CR3 register allows to decrease the current consumption in this mode, but any drop of the voltage below the operating conditions between two active periods of the supply detector results in a non-generation of PDR reset.

#### I/O states in Standby mode

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

The RTC outputs on PC13 and PA4 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. Up to six wake-up pins (WKUPx, x = 1 to 6) and the two tamper pins are available.

#### Entering Standby mode

The device enters Standby mode according to [Entering low-power modes](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M0+ System Control register is set.

Refer to [Table 33: Standby mode summary](#) for details on how to enter Standby mode.

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

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See [Section 28.3: IWDG functional description](#).
- Real-time clock (RTC) and tamper (TAMP): this is configured by the RTCEN bit in the RTC domain control register (RCC\_BDCR)
- Internal RC oscillator (LSI): this is configured by the LSION bit in the Control/status register (RCC\_CSR).
- External 32.768 kHz oscillator (LSE): this is configured by the LSEON bit in the RTC domain control register (RCC\_BDCR)

### Exiting Standby mode

The device exits Standby mode according to section [Entering low-power modes](#). The SBF status flag in the [Power status register 1 \(PWR\\_SR1\)](#) indicates that the device was in Standby mode. All registers are reset after wake-up from Standby except for [Power control register 3 \(PWR\\_CR3\)](#).

Refer to [Table 33: Standby mode summary](#) for more details on how to exit Standby mode.

**Table 33. Standby mode summary**

Characteristic	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ System Control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS = “011” in <a href="#">Power control register 1 (PWR_CR1)</a></li> <li>– WUFx bits are cleared in <a href="#">Power status register 1 (PWR_SR1)</a></li> </ul>
	On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ System Control register</li> <li>– SLEEPONEXIT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS = “011” in <a href="#">Power control register 1 (PWR_CR1)</a></li> <li>– WUFx bits are cleared in <a href="#">Power status register 1 (PWR_SR1)</a></li> <li>– The RTC flag corresponding to the chosen wake-up source (RTC Alarm A, RTC Alarm B, RTC wake-up, tamper or timestamp flags) is cleared</li> </ul>
Mode exit	WKUPx pin edge, RTC event, TAMP event, external reset on NRST pin, IWDG reset, BOR
Wake-up latency	Reset phase

### 4.3.9 Shutdown mode

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

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

#### I/O states in Shutdown mode

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

The RTC outputs on PC13 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. Up to six wake-up pins (WKUPx, x = 1 to 6) and the two tamper are available.

#### Entering Shutdown mode

The device enters Shutdown mode according to section [Entering low-power modes](#), when the SLEEPDEEP bit in the Cortex<sup>®</sup>-M0+ System Control register is set.

Refer to [Table 34: Shutdown mode summary](#) for details on how to enter Shutdown mode.

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

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

**Caution:** The RTC domain content is lost when  $V_{DD}$  is powered down under Shutdown mode.

### Exiting Shutdown mode

The device exits Shutdown mode according to section [Exiting low-power modes](#). A power-on reset occurs when exiting from Shutdown mode. All registers (except for the ones in the RTC domain), the WUFx bits in PWR\_SR1 register inclusive, are reset upon wake-up from Shutdown.

Refer to [Table 34: Shutdown mode summary](#) for more details on how to exit Shutdown mode.

**Table 34. Shutdown mode summary**

Characteristic	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ system control register</li> <li>– No interrupt (for WFI) or event (for WFE) is pending</li> <li>– LPMS[2:0] = 1XX in <a href="#">Power control register 1 (PWR_CR1)</a></li> <li>– WUFx bits are cleared in <a href="#">Power status register 1 (PWR_SR1)</a></li> </ul>
	On return from ISR while: <ul style="list-style-type: none"> <li>– SLEEPDEEP bit is set in Cortex<sup>®</sup>-M0+ system control register</li> <li>– SLEEPONEXT = 1</li> <li>– No interrupt is pending</li> <li>– LPMS[2:0] = 1XX in <a href="#">Power control register 1 (PWR_CR1)</a></li> <li>– WUFx bits are cleared in <a href="#">Power status register 1 (PWR_SR1)</a></li> <li>– The RTC flag corresponding to the chosen wake-up source (RTC Alarm A, RTC Alarm B, RTC wake-up, tamper or timestamp flags) is cleared</li> </ul>
Mode exit	WKUPx pin edge, RTC event, TAMP event, external reset on NRST pin
Wake-up latency	Reset phase

#### 4.3.10 Auto-wake-up from low-power mode

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

- Low-power 32.768 kHz external crystal oscillator (LSE OSC)  
This clock source provides a precise time base with very low-power consumption.
- Low-power internal RC Oscillator (LSI)  
This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.

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

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

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

## 4.4 PWR registers

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

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

Address offset: 0x00

Reset value: 0x0000 0208

This register is reset after wake-up from Standby mode.

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

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

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

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

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

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

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

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

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

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

Bit 5 **FPD\_LPSLP**: Flash memory powered down during Low-power sleep mode

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

Bit 4 **FPD\_LPRUN**: Flash memory powered down during Low-power run mode

This bit determines whether the flash memory is put in power-down mode or remains in idle mode when the device enters Low-power run mode. The flash memory can be put in power-down mode only when the user code is executed from SRAM.

- 0: Flash memory idle
- 1: Flash memory powered down

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 0 mode
- 001: Stop 1 mode
- 010: Reserved
- 011: Standby mode
- 1xx: Shutdown mode

*Note:* In Standby mode, SRAM contents are retained or not, depending on RRS bit setting in PWR\_CR3.

### 4.4.2 Power control register 2 (PWR\_CR2)

Address offset: 0x04

Reset value: 0x0000 0100

This register is reset when exiting 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.	Res.	PVM_VDDIO2[2:0]			PVMEN DAC	PVDRT[2:0]			PVDFT[2:0]			PVDE
					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 10:8 **PVM\_VDDIO2[2:0]**: V<sub>DDIO2</sub> supply voltage monitoring

This bitfield controls the voltage monitoring of V<sub>DDIO2</sub> with respect to 1.2 V threshold and the IOs in the V<sub>DDIO2</sub> domain.

000: Monitoring disabled; IOs in isolation mode

001: Monitoring enabled; IOs enabled or in isolation mode, according to V<sub>DDIO2</sub> level

110: Monitoring bypassed; IOs enabled

Other: Reserved

*Note: This bitfield is only applicable on STM32G0B1xx and STM32G0C1xx. It is reserved on the other products.*

Bit 7 **PVMENDAC**: DAC supply voltage monitoring enable

This bit enables the monitoring of the DAC supply with respect to 1.8 V threshold.

0: Disable

1: Enable

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

These bits select the PVD rising threshold:

000: V<sub>PVDR0</sub> (around 2.1 V)

001: V<sub>PVDR1</sub> (around 2.2 V)

010: V<sub>PVDR2</sub> (around 2.5 V)

011: V<sub>PVDR3</sub> (around 2.6 V)

100: V<sub>PVDR4</sub> (around 2.7 V)

101: V<sub>PVDR5</sub> (around 2.9 V)

110: V<sub>PVDR6</sub> (around 3.0 V)

111: PVD\_IN pin voltage

*Note: If this bitfield is set to 111, the voltage on PVD\_IN pin is internally compared with V<sub>REFINT</sub> for both rising and falling threshold and the PVDFT[2:0] bitfield has no effect.*

*Note: These bits are write-protected when the PVD\_LOCK bit is set in the SYSCFG\_CFGR2 register. The protection can be reset only by a system reset.*

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

These bits select the PVD falling threshold:

000: V<sub>PVDF0</sub> (around 2.0 V)

001: V<sub>PVDF1</sub> (around 2.2 V)

010: V<sub>PVDF2</sub> (around 2.4 V)

011: V<sub>PVDF3</sub> (around 2.5 V)

100: V<sub>PVDF4</sub> (around 2.6 V)

101: V<sub>PVDF5</sub> (around 2.8 V)

110: V<sub>PVDF6</sub> (around 2.9 V)

111: Not used

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

*Note: These bits are write-protected when the PVD\_LOCK bit is set in the SYSCFG\_CFGR2 register. The protection can be reset only by a system reset.*

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

0: Programmable voltage detector disable.

1: Programmable voltage detector enable.

*Note: This bit is write-protected when the PVD\_LOCK bit is set in the SYSCFG\_CFGR2 register. The protection can be reset only by a system reset.*

### 4.4.3 Power control register 3 (PWR\_CR3)

Address offset: 0x08

Reset value: 0x0000 8000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBRRSTR1\)](#).

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	ENB_ULP	RRS	Res.	Res.	EWUP 6	EWUP 5	EWUP 4	EWUP 3	EWUP 2	EWUP 1
rw					rw	rw	rw			rw	rw	rw	rw	rw	rw

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

Bit 15 **EIWUL**: Enable internal wake-up line  
 0: Disable  
 1: Enable

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

Bit 10 **APC**: Apply pull-up and pull-down configuration  
 This bit determines whether the I/O pull-up and pull-down configurations defined in the PWR\_PUCRx and PWR\_PDCRx registers are applied.  
 0: Not applied  
 1: Applied

Bit 9 **ENB\_ULP**: Ultra-low-power enable  
 Enable/disable periodical sampling of supply voltage in Stop and Standby modes for detecting condition of PDR and BOR reset.  
 0: Disable (the supply voltage is monitored continuously)  
 1: Enable  
 When set, the supply voltage is sampled for PDR/BOR reset condition only periodically and not continuously, in order to save power.

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

Bit 8 **RRS**: SRAM retention in Standby mode  
 The bit determines whether the SRAM is powered (from low-power regulator) and retains its contents, or unpowered and loses its contents.  
 0: Unpowered - contents lost  
 1: Powered - contents retained

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



- Bit 5 **EWUP6**: Enable WKUP6 wake-up pin  
 When this bit is set, the WKUP6 external wake-up pin is enabled and triggers a wake-up from Standby or Shutdown mode when a rising or a falling edge occurs. The active edge is configured through WP6 bit in the PWR\_CR4 register.
- Bit 4 **EWUP5**: Enable WKUP5 wake-up pin  
 When this bit is set, the WKUP5 external wake-up pin is enabled and triggers a wake-up from Standby or Shutdown mode when a rising or a falling edge occurs. The active edge is configured via the WP5 bit in the PWR\_CR4 register.
- Bit 3 **EWUP4**: Enable WKUP4 wake-up pin  
 When this bit is set, the WKUP4 external wake-up pin is enabled and triggers a wake-up from Standby or Shutdown mode 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 wake-up pin  
 When this bit is set, the WKUP3 external wake-up pin is enabled and triggers a wake-up from Standby or Shutdown mode 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 wake-up pin  
 When this bit is set, the WKUP2 external wake-up pin is enabled and triggers a wake-up from Standby or Shutdown mode 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 wake-up pin  
 When this bit is set, the WKUP1 external wake-up pin is enabled and triggers a wake-up from Standby or Shutdown mode when a rising or a falling edge occurs. The active edge is configured via the WP1 bit of the PWR\_CR4 register.

#### 4.4.4 Power control register 4 (PWR\_CR4)

Address offset: 0x0C

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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.	VBRS	VBE	Res.	Res.	WP6	WP5	WP4	WP3	WP2	WP1
						rw	rw			rw	rw	rw	rw	rw	rw

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

Bit 9 **VBRS**:  $V_{BAT}$  battery charging resistor selection

0: 5 k $\Omega$

1: 1.5 k $\Omega$

Bit 8 **VBE**:  $V_{BAT}$  battery charging enable

0: Disable

1: Enable

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

Bit 5 **WP6**: WKUP6 wake-up pin polarity

WKUP6 external wake-up signal polarity (level or edge) triggering wake-up event:

0: High level or rising edge

1: Low level or falling edge

Bit 4 **WP5**: WKUP5 wake-up pin polarity

WKUP5 external wake-up signal polarity (level or edge) triggering wake-up event:

0: High level or rising edge

1: Low level or falling edge

Bit 3 **WP4**: WKUP4 wake-up pin polarity

WKUP4 external wake-up signal polarity (level or edge) triggering wake-up event:

0: High level or rising edge

1: Low level or falling edge

Bit 2 **WP3**: WKUP3 wake-up pin polarity

WKUP3 external wake-up signal polarity (level or edge) triggering wake-up event:

0: High level or rising edge

1: Low level or falling edge

Bit 1 **WP2**: WKUP2 wake-up pin polarity

WKUP2 external wake-up signal polarity (level or edge) triggering wake-up event:

0: High level or rising edge

1: Low level or falling edge

Bit 0 **WP1**: WKUP1 wake-up pin polarity

WKUP1 external wake-up signal polarity (level or edge) triggering wake-up event:

0: High level or rising edge

1: Low level or falling edge

#### 4.4.5 Power status register 1 (PWR\_SR1)

Address offset: 0x10

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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
WUFI	Res.	Res.	Res.	Res.	Res.	Res.	SBF	Res.	Res.	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
r							r			r	r	r	r	r	r

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

Bit 15 **WUFI**: Wake-up flag internal

This bit is set when a wake-up is detected on the internal wake-up line. It is cleared when all internal wake-up sources are cleared.

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**: Wake-up flag 6

This bit is set when a wake-up event is detected on WKUP6 wake-up pin. It is cleared by writing 1 in the CWUF6 bit of the PWR\_SCR register.

Bit 4 **WUF5**: Wake-up flag 5

This bit is set when a wake-up event is detected on WKUP5 wake-up pin. It is cleared by writing 1 in the CWUF5 bit of the PWR\_SCR register.

Bit 3 **WUF4**: Wake-up flag 4

This bit is set when a wake-up event is detected on WKUP4 wake-up pin. It is cleared by writing 1 in the CWUF4 bit of the PWR\_SCR register.

Bit 2 **WUF3**: Wake-up flag 3

This bit is set when a wake-up event is detected on WKUP3 wake-up pin. It is cleared by writing 1 in the CWUF3 bit of the PWR\_SCR register.

Bit 1 **WUF2**: Wake-up flag 2

This bit is set when a wake-up event is detected on WKUP2 wake-up pin. It is cleared by writing 1 in the CWUF2 bit of the PWR\_SCR register.

Bit 0 **WUF1**: Wake-up flag 1

This bit is set when a wake-up event is detected on WKUP1 wake-up pin. It is cleared by writing 1 in the CWUF1 bit of the PWR\_SCR register.

#### 4.4.6 Power status register 2 (PWR\_SR2)

Address offset: 0x14

Reset value: 0b0000 0000 0000 0000 X00X XXXX X000 0000 (The bits in this register reflect the actual status.)

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
PVMO DAC	Res.	PVMO VDDIO 2	Res.	PVDO	VOSF	REGLP F	REGLP S	FLASH _RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r		r		r	r	r	r	r							

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

Bit 15 **PVMODAC**:  $V_{DDA}$  monitoring output flag

This flag indicates the readiness of the  $V_{DDA}$  supply voltage (excess of PVM threshold of about 1.8 V).

0:  $V_{DDA}$  ready

1:  $V_{DDA}$  not ready

The flag is cleared when the PVM of the DAC is disabled (PVMENDAC = 0).

Bit 14 Reserved, must be kept at reset value.

Bit 13 **PVMOVDDIO2**:  $V_{DDIO2}$  supply voltage monitoring output flag

This flag indicates the readiness of the  $V_{DDIO2}$  supply voltage (excess of 1.2 V).

0:  $V_{DDIO2}$  ready

1:  $V_{DDIO2}$  not ready

The flag is cleared when the PVM of  $V_{DDIO2}$  is disabled (PVM\_VDDIO2[0] = 0).

*Note: This bitfield is only applicable on STM32G0B1xx and STM32G0C1xx. It is reserved on the other products.*

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PVDO**: Programmable voltage detector output

0:  $V_{DD}$  is above the selected PVD threshold

1:  $V_{DD}$  is below the selected PVD threshold

Bit 10 **VOSF**: Voltage scaling flag

A delay is required for the internal regulator to be ready after the voltage scaling has been changed. VOSF indicates that the regulator reached the voltage level defined with VOS bits of the PWR\_CR1 register.

0: The regulator is ready in the selected voltage range

1: The regulator output voltage is changing to the required voltage level

Bit 9 **REGLPF**: Low-power regulator flag

This flag is controlled by hardware. It indicates the regulator from which the core domain is currently supplied. The core domain is supplied from the low-power regulator (LPR) while the device is in Low-power run mode. Upon transiting to Run mode, it remains supplied from the LPR until the main regulator (MR) becomes ready. Poll this flag before increasing the product frequency.

0: The core domain is supplied from the main regulator (MR)

1: The core domain is supplied from the low-power regulator (LPR)

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

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

0: The low-power regulator is not ready

1: The low-power regulator is ready

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

This bit is set by hardware to indicate when the flash memory is ready to be accessed after wake-up from power-down. To place the flash memory in power-down, set either FPD\_LPRUN, FPD\_LPSTP or FPD\_STOP bits.

0: Flash memory in power-down

1: Flash memory ready to be accessed

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

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

### 4.4.7 Power status clear register (PWR\_SCR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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.	CSBF	Res.	Res.	CWUF 6	CWUF 5	CWUF 4	CWUF 3	CWUF 2	CWUF 1
							w			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 wake-up flag 6

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

Bit 4 **CWUF5**: Clear wake-up flag 5

Setting this bit clears the WUF5 flag in the PWR\_SR1 register.

Bit 3 **CWUF4**: Clear wake-up flag 4

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

- Bit 2 **CWUF3**: Clear wake-up flag 3  
Setting this bit clears the WUF3 flag in the PWR\_SR1 register.
- Bit 1 **CWUF2**: Clear wake-up flag 2  
Setting this bit clears the WUF2 flag in the PWR\_SR1 register.
- Bit 0 **CWUF1**: Clear wake-up flag 1  
Setting this bit clears the WUF1 flag in the PWR\_SR1 register.

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

Address offset: 0x20

Reset value: 0x0000 0000

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

- Bits 15:0 **PUy**: Port A pull-up bit y (y = 15 to 0)  
Setting PUy bit while the corresponding PDy bit is zero and the APC bit of the PWR\_CR3 register is set activates a pull-up device on the PA[y] I/O.

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

Address offset: 0x24

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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

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

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

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

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

Address offset: 0x28

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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

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

Bits 15:0 **PUy**: Port B pull-up bit y (y = 15 to 0)

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

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

Address offset: 0x2C

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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

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

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

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

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

Address offset: 0x30

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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

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

Bits 15:0 **PUy**: Port C pull-up bit y (y = 15 to 0)<sup>(1)</sup>

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

1. In STM32G031xx and STM32G041xx as well as STM32G051xx and STM32G061xx devices, the bits PD0 to PD5 and PD8 to PD12 are reserved.

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

Address offset: 0x34

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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



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

Bits 15:0 **PDy**: Port C pull-down bit y (y = 15 to 0)<sup>(1)</sup>

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

1. In STM32G031xx and STM32G041xx as well as STM32G051xx and STM32G061xx devices, the bits PD0 to PD5 and PD8 to PD12 are reserved.

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

Address offset: 0x38

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBRRSTR1\)](#).

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

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

Bits 15:0 **PUy**: Port D pull-up bit y (y = 15 to 0)<sup>(1)</sup>

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

1. In STM32G071xx and STM32G081xx devices, the bits PU15 to PU10 and PU7 are reserved. In STM32G031xx and STM32G041xx as well as in STM32G051xx and STM32G061xx devices, PU15 to PU4 are reserved.

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

Address offset: 0x3C

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBRRSTR1\)](#).

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

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

Bits 15:0 **PDy**: Port D pull-down bit y (y = 15 to 0)<sup>(1)</sup>

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

1. In STM32G071xx and STM32G081xx devices, the bits PD15 to PD10 and PD7 are reserved. In STM32G031xx and STM32G041xx as well as in STM32G051xx and STM32G061xx devices, PD15 to PD4 are reserved.

#### 4.4.16 Power Port E pull-up control register (PWR\_PUCRE)

Address offset: 0x40

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWR\_RST bit of the [APB peripheral reset register 1 \(RCC\\_APBSTR1\)](#).

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

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

Bits 15:0 **PUy**: Port E pull-up bit y (y = 15 to 0)<sup>(1)</sup>

Setting PUy bit while the corresponding PDy bit is zero and the APC bit of the PWR\_CR3 register is set activates a pull-up device on the PE[y] I/O.

1. Only applies to STM32G0B1xx and STM32G0C1xx devices. Reserved for the other devices.

#### 4.4.17 Power Port E pull-down control register (PWR\_PDCRE)

Address offset: 0x44

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBRRSTR1\)](#).

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

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

Bits 15:0 **PDy**: Port E pull-down bit y (y = 15 to 0)<sup>(1)</sup>

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

1. Only applies to STM32G0B1xx and STM32G0C1xx devices. Reserved for the other devices.

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

Address offset: 0x48

Reset value: 0x0000 0000

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

This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the [APB peripheral reset register 1 \(RCC\\_APBRRSTR1\)](#).

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.	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
		rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bits 13:0 **PUy**: Port F pull-up bit y (y = 13 to 0)<sup>(1)</sup>

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

1. Bits PU13 to PU3 only applies to STM32G0B1xx and STM32G0C1xx devices. Reserved for the other devices.

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

Address offset: 0x4C

Reset value: 0x0000 0000

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



This register is neither reset upon exiting Standby mode nor with the PWRRST bit of the *APB peripheral reset register 1 (RCC\_APBRRSTR1)*.

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

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

Bits 13:0 **PDy**: Port F pull-down bit y (y = 13 to 0)<sup>(1)</sup>

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

1. Bits PU13 to PU3 only applies to STM32G0B1xx and STM32G0C1xx devices. Reserved for the other devices.

4.4.20 PWR register map

Table 35. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS [1:0]	DBP	Res.	Res.	Res.	FPD_LPSLP	FPD_LPRUN	FPD_STOP	LPMS [2:0]					
	Reset value																			0	Res.	Res.	0	1	0			0	0	1	0	0	0			
0x004	PWR_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USV	IOSV	PVMENUSB	PVMENDAC	Res.	PVDRT [2:0]		PVDFT [2:0]		PVDE				
	Reset value																						0	0	0	0	0	0	0	0	0	0	0			
0x008	PWR_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC	ENB_ULP	RRS	Res.	Res.	Res.	EWUP6	EWUP5	EWUP4	Res.	EWUP2	EWUP1		
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00C	PWR_CR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	VBE	Res.	Res.	WP6	WP5	WP4	Res.	WP2	WP1			
	Reset value																							0	0	0	0	0	0	0	0	0	0	0	0	
0x010	PWR_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	PWR_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	PWR_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																			
0x020	PWR_PUCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	PWR_PDCRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																			
0x028	PWR_PUCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																			
0x02C	PWR_PDCRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	PWR_PUCRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	PWR_PDCRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 35. PWR register map and reset values (continued)**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x038	PWR_PUCRD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x03C	PWR_PDCRD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x040	PWR_PUCRE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x044	PWR_PDCRE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x048	PWR_PUCRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x04C	PWR_PDCRF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 60](#) 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)
- Exit from Standby mode
- Exit from Shutdown mode

Power reset sets all registers to their reset values except the registers of the RTC domain.

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

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

#### 5.1.2 System reset

A system reset sets all registers to their reset value unless otherwise specified in the register descriptions.

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

- Low level on NRST (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 (POR) or brown-out reset (BOR)

The reset source can be identified by checking the reset flags in the RCC\_CSR register (see [Section 5.4.24: Control/status register \(RCC\\_CSR\)](#)).

##### NRST (external reset):

Through specific option bits, the PF2-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  $\mu$ s for each internal reset source to be output on the NRST pin. An internal reset holder option can be used, if enabled in the option bytes, to ensure that the pin is pulled low until it is read as low level. This function allows the detection of internal reset sources by external components when the line faces a significant

capacitive load. The BOOT0 pin is sampled on any NRST rising edge, caused by internal or external resets.

- **Reset input**

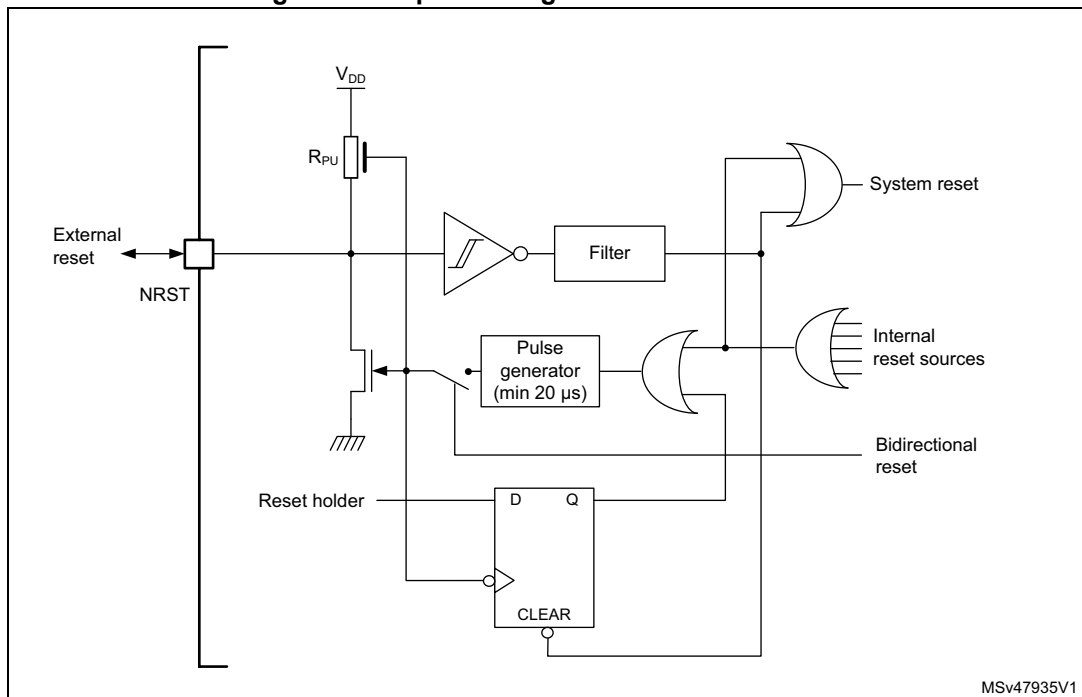
In this mode, any valid reset signal on the NRST pin is propagated to device internal logic, but resets generated internally by the device are not visible on the pin. The GPIO functionality (PF2) is not available. The BOOT0 pin is sampled on POR and any subsequent NRST rising edge, caused by external resets. Other internal resets do not trigger new BOOT0 sampling.

- **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. The BOOT0 pin is sampled on POR NRST rising edge only. Subsequent internal resets or transitions on the PF2 GPIO do not trigger new BOOT0 sampling. Also refer to [Section 7.3.17: Reset pin \(PF2-NRST\) in GPIO mode](#) for additional considerations for this mode.

**Caution:** Upon power-on reset or wake-up from Standby or Shutdown mode, the PF2-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  $t_{RSTEMPO}$ .

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



**Software reset**

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



### Low-power mode security reset

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

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

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

### Option byte loader reset

The option byte loader reset is generated when the OBL\_LAUNCH bit (bit 27) is set in the FLASH\_CR register. This bit is used to launch the option byte loading by software.

## 5.1.3 RTC domain reset

The RTC domain has two specific resets.

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

- **Software reset**, triggered by setting the BDRST bit in the [RTC domain control register \(RCC\\_BDCR\)](#).
- **V<sub>DD</sub> or V<sub>BAT</sub> power on**, if both supplies have previously been powered off.

A RTC domain reset only affects the LSE oscillator, the RTC, the backup registers and the RCC RTC domain control register.

## 5.2 Clocks

The device provides the following clock sources producing primary clocks:

- **HSI16 RC** - a high-speed fully-integrated RC oscillator producing **HSI16** clock (about 16 MHz)
- **HSI48 RC** - a high-speed fully-integrated RC oscillator producing **HSI48** clock for USB (about 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 or/and multiplying the primary clocks:

- **HSISYS** - a clock derived from HSI16 through division by a factor programmable from 1 to 128
- **PLLCLK, PLLQCLK and PLLRCLK** - clocks output from the PLL block
- **SYSCCLK** - a clock obtained through selecting one of LSE, LSI, HSE, PLLRCLK, and HSISYS clocks
- **HCLK** - a clock derived from SYSCCLK through division by a factor programmable from 1 to 512
- **HCLK8** - a clock derived from HCLK through division by eight
- **PCLK** - a clock derived from HCLK through division by a factor programmable from 1 to 16
- **TIMPCLK** - a clock derived from PCLK, running at PCLK frequency if the APB prescaler division factor is set to 1, or at twice the PCLK frequency otherwise
- **LPTIMx\_IN** - clock from LPTIMx\_INx pins, selectable for the LPTIM peripheral

More secondary clocks are generated by fixed division of HSE, HSI16 and HCLK clocks.

The HSISYS is used as system clock source after startup from reset, with the division by 1 (producing HSI16 frequency).

The HCLK clock and PCLK clock are used for clocking the AHB and the APB domains, respectively. Their maximum allowed frequency is 64 MHz.

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

- **TIMx**, with these clock sources to select from:
  - TIMPCLK (selectable for all timers) running at PCLK frequency if the APB prescaler division factor is set to 1, or at twice the PCLK frequency otherwise
  - PLLQCLK selectable for high-speed TIM1 and TIM15 timers
- **LPTIMx**, with these clock sources to select from:
  - LSI
  - LSE
  - HSI16
  - PCLK (APB clock)
  - LPTIMx\_IN selected from LPTIMx\_INx pins

The functionality in Stop mode (including wake-up) is supported only when the clock is LSI or LSE.

- **UCPD**, always clocked with HSI16
- **ADC**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - PLLPCLK
- **USARTx / LPUARTx**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - LSE
  - PCLK (APB clock)

The wake-up from Stop mode is supported only when the clock is HSI16 or LSE.

- **I2Cx**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - PCLK (APB clock)

The wake-up from Stop mode is supported only when the clock is HSI16.

- **I2Sx**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16
  - PLLPCLK
  - I2S\_CKIN pin

- **RNG**, with these clock sources to select from:
  - SYSCLK (system clock)
  - HSI16 clock divided by 8
  - PLLQCLK

The RNG clock can additionally be divided by 2, 4 or 8, using a dedicated prescaler.

- **CEC**, with these clock sources to select from:
  - HSI16 clock divided by 488
  - LSE
- **RTC**, with these clock sources to select from:
  - LSE
  - LSI
  - HSE clock divided by 32

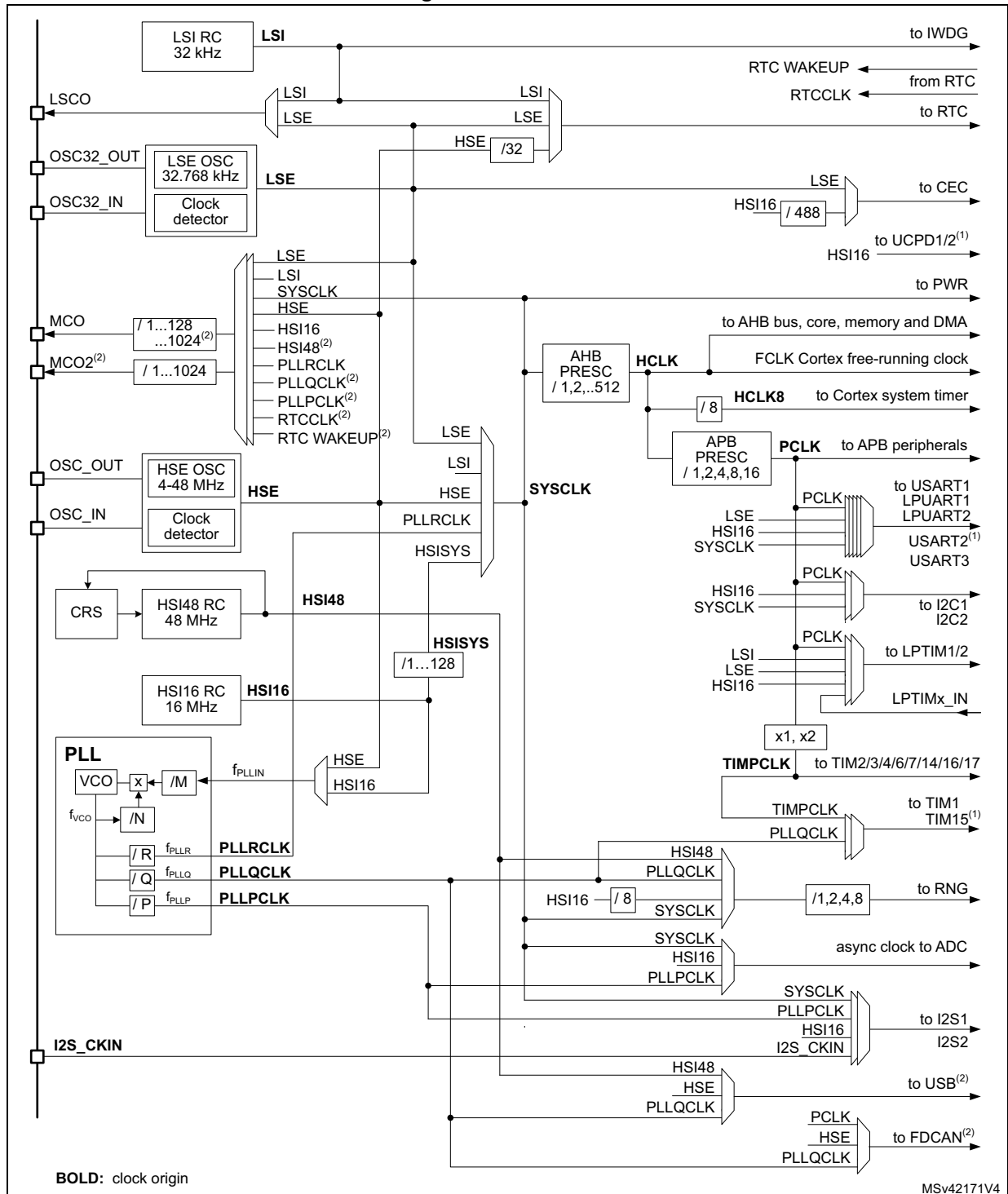
The functionality in Stop mode (including wake-up) is supported only when the clock is LSI or LSE.

- **IWDG**, always clocked with LSI clock.
- **USB**, with these clocks to select from:
  - HSE
  - HSI48
  - PLLQCLK
- **FDCAN**, with these clocks to select from:
  - HSE
  - PCLK
  - PLLQCLK
- **SysTick** (Cortex<sup>®</sup> core system timer), with these clock sources to select from:
  - HCLK (AHB clock)
  - HCLK clock divided by 8

The selection is done through SysTick control and status register.

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

Figure 10. Clock tree



1. Only applies to STM32G071xx and STM32G081xx and to STM32G0B1xx and STM32G0C1xx.
2. Only applies to STM32G0B1xx and STM32G0C1xx.

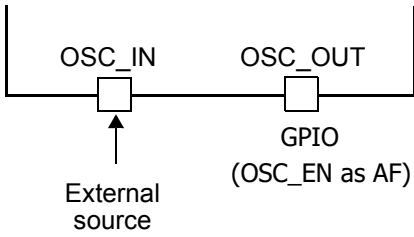
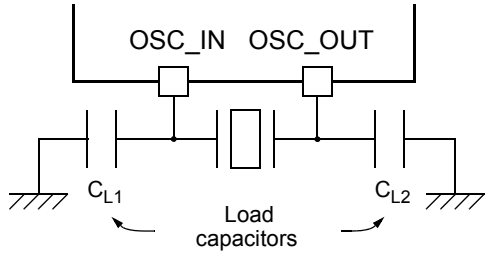
### 5.2.1 HSE clock

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

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

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

**Figure 11. HSE/ LSE clock sources**

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 11](#). Refer to the electrical characteristics section of the *datasheet* for more details.

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

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

### External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 48 MHz. This mode is selected by setting the HSEBYP and HSEON bits in the [Clock control register \(RCC\\_CR\)](#). The external clock signal (square, sinus or triangle) with ~40-60 % duty cycle depending on the frequency (refer to the *datasheet*) must drive the OSC\_IN pin (see [Figure 11](#)).

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. The OSC\_EN output is high when the external HSE clock is required and low when the external HSE clock can be switched off. It allows stopping the external clock source when the device enters low power modes.

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

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

## 5.2.2 HSI16 clock

The HSI16 clock signal is generated from an internal 16 MHz RC oscillator.

The HSI16 RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator. However, even after calibration, it is less accurate than an oscillator using a frequency reference such as quartz crystal or ceramic resonator.

The HSI16 clock derived from HSI16 can be selected as system clock after wake-up from Stop modes (Stop 0 or Stop 1). Refer to [Section 5.3: Low-power modes](#). It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 5.2.9: 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 [Internal clock source calibration register \(RCC\\_ICSCR\)](#).

Voltage or temperature variations in the application may affect the HSI16 frequency of the RC oscillator. It can be trimmed using the HSITRIM[6:0] bits in the [Internal clock source calibration register \(RCC\\_ICSCR\)](#).

For more details on how to measure the HSI16 frequency variation, refer to [Section 5.2.16: Internal/external clock measurement with TIM14/TIM16/TIM17](#).

The HSIRDY flag in the [Clock control register \(RCC\\_CR\)](#) indicates if the HSI16 RC is stable or not. At startup, the HSI16 RC output clock is not released until this bit is set by hardware.

The HSI16 RC can be switched on and off using the HSION bit in the [Clock control register \(RCC\\_CR\)](#).

The HSI16 signal can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 5.2.9: Clock security system \(CSS\) on page 166](#).

### 5.2.3 HSI48 clock

Available on the STM32G0B1xx and STM32G0C1xx devices only, the HSI48 clock signal is generated from an internal 48 MHz RC oscillator. It can be used as clock source for the USB and RNG peripherals.

The internal 48MHz RC oscillator provides a high-precision clock to the USB peripheral thanks to the clock recovery system (CRS). CRS uses the USB SOF signal, LSE clock or an external signal as timing reference to precisely adjust the HSI48 RC oscillator frequency.

HSI48 RC oscillator is disabled as soon as the system enters in Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its free-run frequency which is subject to manufacturing process variations. The devices are factory-calibrated for ~3 % accuracy at  $T_A = 25^\circ\text{C}$ .

Refer to CRS section for more details on how to configure and use CRS.

The HSI48RDY flag in the RCC\_CR register indicates if HSI48 is stable or not. At startup, the HSI48 clock is not released until this flag is set by hardware.

The HSI48 RC oscillator is enabled/disabled through the HSI48ON bit of the RCC\_CR register. It is automatically enabled (by hardware setting the HSI48ON bit) when selected as clock source for the USB peripheral, as long as the USB peripheral is enabled.

Furthermore, it is possible to output the HSI48 clock through the MCO and MCO2 multiplexers and use it as a clock source for other application components.

### 5.2.4 PLL

The internal PLL multiplies the frequency of HSI16- or HSE-based clock fetched on its input, to produce three independent clock outputs. The allowed input frequency range is from 2.66 to 16 MHz. The dedicated divider PLLM with division factor programmable from one to eight allows setting a frequency within the valid PLL input range. Refer to [Figure 10: Clock tree](#) and [PLL configuration register \(RCC\\_PLLCFGR\)](#).

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

To modify the PLL configuration, proceed as follows:



1. Disable the PLL by setting PLLON to 0 in *Clock control register (RCC\_CR)*.
2. Wait until PLLRDY is cleared. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.
5. Enable the desired PLL outputs by configuring PLLPEN, PLLQEN, and PLLREN in *PLL configuration register (RCC\_PLLCFGR)*.

An interrupt can be generated when the PLL is ready, if enabled in the *Clock interrupt enable register (RCC\_CIER)*.

The enable bit of each PLL output clock (PLLPEN, PLLQEN, and PLLREN) can be modified at any time without stopping the PLL. PLLREN cannot be cleared if PLLRCLK is used as system clock.

### 5.2.5 LSE clock

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

The LSE crystal is switched on and off using the LSEON bit in *RTC domain control register (RCC\_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *RTC domain control register (RCC\_BDCR)* to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV=00) when the LSE is ON. However, once LSEDRV is selected, the drive capability can not be increased if LSEON=1.

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

#### External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the *AHB peripheral clock enable in Sleep/Stop mode register (RCC\_AHBSMENR)*. The external clock signal (square, sinus or triangle) with ~50 % duty cycle has to drive the OSC32\_IN pin while the OSC32\_OUT pin can be used as GPIO. See *Figure 11*.

OSC32\_OUT can be configured as alternate function of OSC32\_EN to control the external clock source. OSC32\_EN output is high when the external LSE clock is required and low when the external LSE clock can be switched off.

### 5.2.6 LSI clock

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

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

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

### 5.2.7 System clock (SYSCLK) selection

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

- LSI
- LSE
- HSISYS
- HSE
- PLLRCLK

The system clock maximum frequency is 64 MHz. Upon system reset, the HSISYS clock derived from HSI16 oscillator is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *Clock control register (RCC\_CR)* indicate which clocks are ready. Status bits in the *Clock configuration register (RCC\_CFGR)* indicate which clock is currently used as a system clock.

### 5.2.8 Clock source frequency versus voltage scaling

The following table gives the different clock source frequencies depending on the product voltage range.

**Table 36. Clock source frequency**

Clock	Maximum clock frequency (MHz)	
	Range 1	Range 2
HSI16	16	16
HSE	48	16
HSI48	48	N/A
PLLCLK	122 <sup>(1)</sup>	40 <sup>(2)</sup>
PLLQCLK	128 <sup>(1)</sup>	32 <sup>(2)</sup>
PLLRCLK	64 <sup>(1)</sup>	16 <sup>(2)</sup>

1. Maximum VCO frequency is 344 MHz.

2. Maximum VCO frequency is 128 MHz.

### 5.2.9 Clock security system (CSS)

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

If a failure is detected on the HSE clock:

- the HSE oscillator is automatically disabled
- a clock failure event is sent to the break input of TIM1, TIM15, TIM16 and TIM17 timers
- CSSI (clock security system interrupt) is generated

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

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

If HSE is selected directly or indirectly (PLLCLK selected for SYSCLK and HSE selected as PLL input) as system clock, and a failure of HSE clock is detected, the system clock switches automatically to HSI and the HSE oscillator is disabled. If the HSE clock (divided or not) is the clock entry of the PLL and PLLCLK is used as system clock when the failure occurs, the PLL is disabled, too.

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

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

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

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

If LSE is used as system clock, and a failure of LSE clock is detected, the system clock switches automatically to LSI. In low-power modes, an LSE clock failure generates a wake-up. 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.

**Caution:** When the LSECSSD bit is set, the LSE oscillator is held under reset and it cannot be restarted. To clear the LSECSSD bit, reset the RTC domain, using BDRST bit.

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

### 5.2.11 ADC clock

The ADC clock is derived from the system clock, or from the PLLPCLK output. It can reach 122 MHz and can be divided by the following prescalers values:

1,2,4,6,8,10,12,16,32,64,128 or 256 by configuring the ADC1\_CCR register. It is asynchronous to the AHB clock. Alternatively, the ADC clock can be derived from the AHB

clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). This programmable factor is configured using the CKMODE bitfields in the ADC1\_CCR.

If the programmed factor is 1, the AHB prescaler must be set to 1.

### 5.2.12 RTC clock

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

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

- If LSE is selected as RTC clock:
  - The RTC continues to work even if the  $V_{DD}$  supply is switched off, provided the  $V_{BAT}$  supply is maintained.
- If LSI is selected as the RTC clock:
  - The RTC state is not guaranteed if the  $V_{DD}$  supply is powered off.
- If the HSE clock divided by a prescaler is used as the RTC clock:
  - The RTC state is not guaranteed if the  $V_{DD}$  supply is powered off or if the internal voltage regulator is powered off (removing power from the  $V_{CORE}$  domain).

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

### 5.2.13 Timer clock

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

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

For TIM1 and TIM15, PLLQCLK clock can also be selected, if:

- PCLK is derived from PLLRCLK, and
- PLLQCLK frequency is an integer multiplication by 2 or more of the PCLK frequency, without exceeding 128 MHz.

### 5.2.14 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.

## 5.2.15 Clock-out capability

### MCO and MCO2

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

- LSI
- LSE
- SYSCLK
- HSI16
- HSI48
- HSE
- PLLPCLK
- PLLQCLK
- PLLRCLK
- RTCCLK
- RTC WAKEUP

The multiplexers for MCO and MCO2, respectively, are controlled by the MCOSEL[3:0] and MCO2SEL[3:0] bitfields of the [Clock configuration register \(RCC\\_CFGR\)](#). Their outputs are further divided by a factor set through the MCOPRE[3:0] and MCO2PRE[3:0] bitfields of the [Clock configuration register \(RCC\\_CFGR\)](#).

*Note:* The MCO2 output and the associated MCO2SEL[3:0] and MCO2PRE[3:0] bitfields are only available on the STM32G0B1xx and STM32G0C1xx devices.

*The divider ratio (MCOPRE[3:0] bitfield) on the devices other than STM32G0B1xx and STM32G0C1xx is limited to 128. The selectable clock sources here do not include PLLPCLK, PLLQCLK, HSI48, RTCCLK, and RTC WAKEUP.*

### LSCO

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

- LSI
- LSE

The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN bit of the [RTC domain control register \(RCC\\_BDCR\)](#). When this additional function is activated on a GPIO, its input buffer is disabled by hardware.

This function remains available in Stop 0, Stop 1 and Standby modes.

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

It is possible to indirectly measure the frequency of all on-board clock sources with the TIM14, TIM16 and TIM17 channel 1 input capture, as represented in [Figure 12](#), [Figure 13](#) and [Figure 14](#).

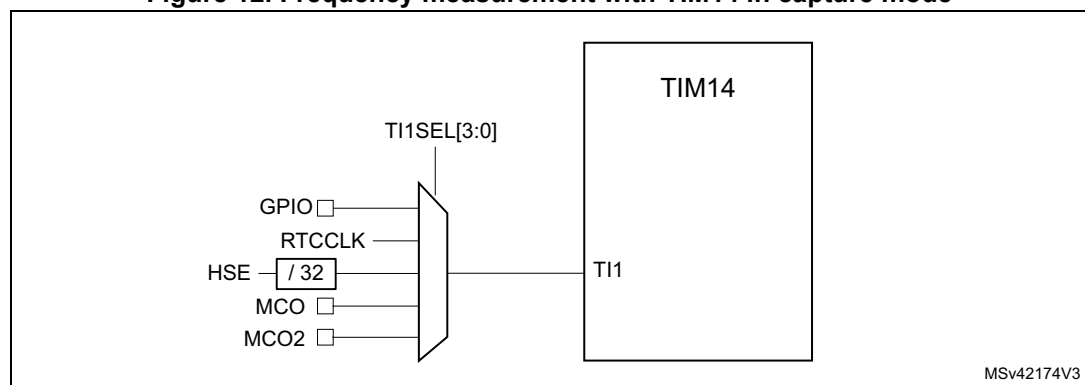
## TIM14

By setting the TI1SEL[3:0] field of the TIM14\_TISEL register, the clock selected for the input capture 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 2)

The two last options are controlled by the MCOSEL[3:0] and MCO2SEL[3:0] bitfields of the [Clock configuration register \(RCC\\_CFGR\)](#). All clock sources can be selected for the MCO and MCO2 pins.

**Figure 12. Frequency measurement with TIM14 in capture mode**



## TIM16

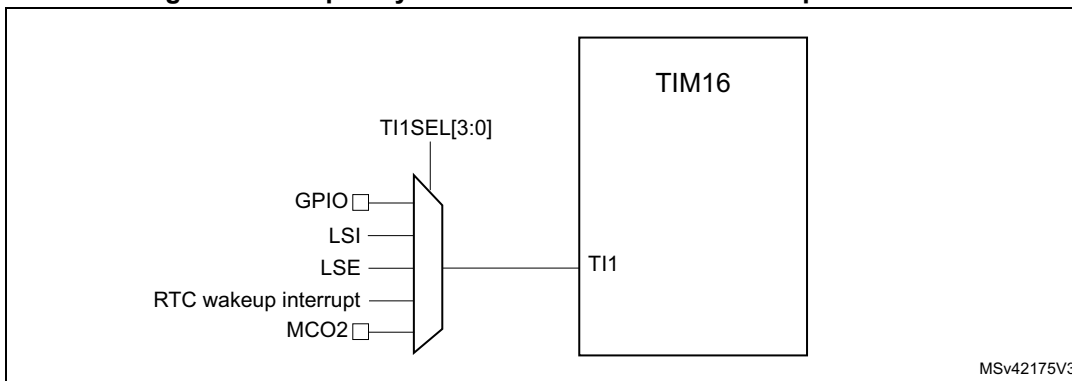
By setting the TI1SEL[3:0] field of the TIM16\_TISEL register, the clock selected for the input capture channel1 of TIM16 can be one of:

- GPIO (refer to the alternate function mapping in the device datasheets).
- LSI clock
- LSE clock
- RTC wake-up interrupt signal
- MCO2 (MCU clock output 2)

The second-last option requires to enable the RTC interrupt.

The last option is controlled through the MCO2SEL[3:0] bitfield of the [Clock configuration register \(RCC\\_CFGR\)](#). All clock sources can be selected for the MCO2 pin.

Figure 13. Frequency measurement with TIM16 in capture mode



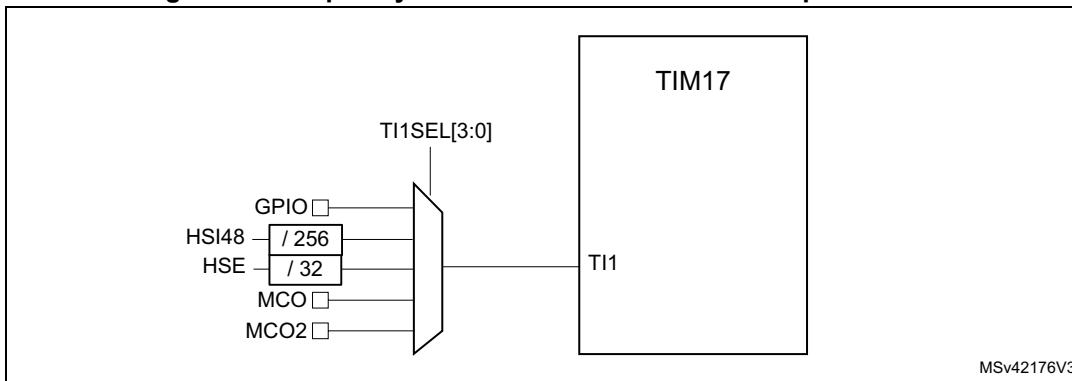
**TIM17**

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)
- HSI48 divided by 256
- HSE divided by 32
- MCO (MCU clock output)
- MCO2 (MCU clock output 2)

The two last options are controlled by the MCOSEL[3:0] and MCO2SEL[3:0] bitfields of the [Clock configuration register \(RCC\\_CFGR\)](#). All clock sources can be selected for the MCO and MCO2 pins.

Figure 14. Frequency measurement with TIM17 in capture mode



**Calibration of the HSI16 oscillator**

For TIM14, TIM15 and TIM17, the primary purpose of connecting the LSE to the channel 1 input capture is to precisely measure HSISYS (derived from HSI16) selected as system clock. Counting HSISYS clock pulses between consecutive edges of the LSE clock (the time reference) allows measuring the HSISYS (and HSI16) clock period. Such measurement can determine the HSI16 oscillator frequency with nearly the same accuracy as the accuracy of the 32.768 kHz quartz crystal used with the LSE oscillator (typically a few tens of ppm). The HSI16 oscillator can then be trimmed to compensate for deviations from target frequency, due to manufacturing, process, temperature and/or voltage variation.

The HSI16 oscillator has dedicated user-accessible calibration bits for this purpose.

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

Generated by the HSE oscillator, the HSE clock (divided by 32) used as time reference is the second best method for reaching a good HSI16 frequency measurement accuracy. It is recommended in absence of the LSE clock.

In order to further improve the precision of the HSI16 oscillator calibration, it is advised to employ one or a combination of the following measures to increase the frequency measurement accuracy:

- set the HSISYS divider to 1 for HSISYS frequency to be equal to HSI16 frequency
- average the results of multiple consecutive measurements
- use the input capture prescaler of the timer (one capture every up to eight periods)
- use LSE clock for the RTC and the RTC wake-up interrupt signal as time reference

The last point significantly increases the reference period for HSI16 clock pulse counting, which improves the accuracy of a single measurement. For operation, the RTC wake-up interrupt must be enabled.

#### Calibration of the HSI48 oscillator

The HSI48 oscillator is factory-calibrated.

#### Measurement of the LSI oscillator frequency

The measurement of the LSI oscillator frequency uses the same principle as that for calibrating the HSI16 oscillator. TIM16 channel1 input capture must be used for LSI clock, and HSE selected as system clock source. The number of HSE clock pulses between consecutive edges of the LSI signal, counted by TIM16, is then representative of the LSI clock period.

### 5.2.17 Peripheral clock enable registers

The clock to each peripheral can individually be enabled by the corresponding enable bit of the RCC\_AHBENR register or one of the RCC\_APBENRx registers. The clocks to I/O ports can individually be enabled through the RCC\_IOPENR register.

When the clock to a peripheral or I/O port is not active, the read and write accesses to its registers are not effective.

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

## 5.3 Low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep and Low Power Sleep modes stops the CPU clock. The memory interface clocks (flash memory and SRAM interfaces) can be stopped by software during sleep mode.



The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.

- Stop modes (Stop 0 and Stop 1) stop all the clocks in the  $V_{CORE}$  domain and disable the PLL as well as the HSI16, HSI48 and HSE oscillators.

The USART1, USART2, USART3, LPUART1, LPUART2, I2C1, and I2C2 peripherals can enable the HSI16 oscillator even when the MCU is in Stop mode (if HSI16 is selected as clock source for one of those peripherals).

The LPUART1, LPUART2, USART1, USART2, and USART3 peripherals can also operate with the clock from the LSE oscillator when the system is in Stop mode, if LSE is selected as clock source for that peripheral and the LSE oscillator is enabled (LSEON set). In that case, the LSE oscillator remains active when the device enters Stop mode (these peripherals do not have the capability to turn on the LSE oscillator).

- Standby and Shutdown modes stop all clocks in the  $V_{CORE}$  domain and disable the PLL, as well as the HSI16, HSI48, and HSE oscillators.

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

When leaving the Stop 0 or Stop 1 modes, HSI16 becomes automatically the system clock.

When leaving the Standby and Shutdown modes, HSI16 (with frequency equal to HSI16) becomes automatically the system clock. At wake-up 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 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.

This register only supports word and half-word accesses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PLL RDY	PLLON	HSI48 RDY	HSI48 ON	Res.	Res.	CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw	r	rw			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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	r	rw	rw								

Bits 31:26 Reserved, must be kept at reset value.

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

Set by hardware to indicate that the PLL is locked.

0: PLL unlocked

1: PLL locked

Bit 24 **PLLON**: PLL enable

Set and cleared by software to enable the PLL.

Cleared by hardware when entering Stop, Standby or Shutdown mode. This bit cannot be reset if the PLL clock is used as the system clock.

0: PLL OFF

1: PLL ON

Bit 23 **HSI48 RDY**: HSI48 clock ready flag

The flag is set when the HSI48 clock is ready for use.

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 22 **HSI48 ON**: HSI48 RC oscillator enable

0: Disable

1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bits 21:20 Reserved, must be kept at reset value.

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

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

0: Clock security system OFF (clock detector OFF)

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

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

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

0: HSE crystal oscillator not bypassed

1: HSE crystal oscillator bypassed with external clock

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

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE oscillator not ready

1: HSE oscillator ready

*Note: Once the HSEON bit is cleared, HSERDY goes low after 6 HSE clock cycles.*

**Bit 16 HSEON:** HSE clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop, Standby, or Shutdown mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF

1: HSE oscillator ON

Bits 15:14 Reserved, must be kept at reset value.

**Bits 13:11 HSIDIV[2:0]:** HSI16 clock division factor

This bitfield controlled by software sets division factor of the HSI16 clock divider to produce HSI16 clock:

000: 1

001: 2

010: 4

011: 8

100: 16

101: 32

110: 64

111: 128

**Bit 10 HSIRDY:** HSI16 clock ready flag

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.

0: HSI16 oscillator not ready

1: HSI16 oscillator ready

*Note: Once the HSION bit is cleared, HSIRDY goes low after 6 HSI16 clock cycles.*

**Bit 9 HSIKERON:** HSI16 always enable for peripheral kernels.

Set and cleared by software to force HSI16 ON even in Stop modes. The HSI16 can only feed USART1, USART2, CEC and I2C1 peripherals configured with HSI16 as kernel clock. Keeping the HSI16 ON in Stop mode allows avoiding to slow down the communication speed because of the HSI16 startup time. This bit has no effect on HSION value.

0: No effect on HSI16 oscillator.

1: HSI16 oscillator is forced ON even in Stop mode.

Bit 8 **HSION**: HSI16 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSI16 oscillator when entering Stop, Standby, or Shutdown mode.

Forced by hardware to keep the HSI16 oscillator ON when it is used directly or indirectly as system clock (also when leaving Stop, Standby, or Shutdown modes, or in case of failure of the HSE oscillator used for system clock).

0: HSI16 oscillator OFF

1: HSI16 oscillator ON

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

### 5.4.2 Internal clock source calibration register (RCC\_ICSCR)

Address offset: 0x04

Reset value: 0x0000 40XX (X is factory-programmed)

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]							
	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r	r	r	r	r	r	r	r

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

Bits 14:8 **HSITRIM[6:0]**: HSI16 clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI16 clock.

The default value is 64, which, when added to the HSICAL value, trims the HSI16 to 16 MHz ± 1 %.

Bits 7:0 **HSICAL[7:0]**: HSI16 clock calibration

These bits are initialized at startup with the factory-programmed HSI16 calibration trim value. When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value. Refer to DS for the trimming steps granularity. The frequency progression presents discontinuities when HSICAL crosses multiples of 64.

### 5.4.3 Clock configuration register (RCC\_CFGR)

One or two wait states are inserted if this register is accessed during clock source switch, and between zero and 15 wait states are inserted if during an update of 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
MCOPRE[3:0]				MCOSEL[3:0]				MCO2PRE[3:0]				MCO2SEL[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
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 31:28 **MCOPRE[3:0]**: 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:

- 0000: 1
- 0001: 2
- 0010: 4
- ...
- 0111: 128
- 1000: 256\*
- 1001: 512\*
- 1010: 1024\*
- Other: reserved

It is highly recommended to set this field before the MCO output is enabled.

*Note: The MCOPRE[3] bit and so the values marked with \* are only significant for STM32G0B1xx and STM32G0C1xx. Reserved on the other devices.*

Bits 27:24 **MCOSEL[3:0]**: Microcontroller clock output clock selector

This bitfield is controlled by software. It sets the clock selector for MCO output as follows:

- 0000: no clock
- 0001: SYSCLK
- 0010: HSI48\*
- 0011: HSI16
- 0100: HSE
- 0101: PLLRCLK
- 0110: LSI
- 0111: LSE
- 1000: PLLPCLK\*
- 1001: PLLQCLK\*
- 1010: RTCCLK\*
- 1011: RTC wake-up\*
- Other: no clock

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

*Values marked with \* are only significant for STM32G0B1xx and STM32G0C1xx. Reserved for the other devices.*

Bits 23:20 **MCO2PRE[3:0]**: 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:

0000: 1

0001: 2

0010: 4

...

0111: 128

1000: 256

1001: 512

1010: 1024

Other: reserved

It is highly recommended to set this field before the MCO2 output is enabled.

*Note: These bits are only significant on devices integrating the corresponding output, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bits 19:16 **MCO2SEL[3:0]**: Microcontroller clock output 2 clock selector

This bitfield is controlled by software. It sets the clock selector for MCO2 output as follows:

0000: no clock

0001: SYSCLK

0010: HSI48

0011: HSI16

0100: HSE

0101: PLLRCLK

0110: LSI

0111: LSE

1000: PLLPCLK

1001: PLLQCLK

1010: RTCCLK

1011: RTC wake-up

Other: no clock

*Note: This clock output may have some truncated cycles at startup or during MCO2 clock source switching.*

*Note: These bits are only significant on devices integrating the corresponding output, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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:

0xxx: 1  
 1000: 2  
 1001: 4  
 1010: 8  
 1011: 16  
 1100: 64  
 1101: 128  
 1110: 256  
 1111: 512

**Caution:** Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details, refer to [Section 4.1.4: Dynamic voltage scaling management](#)). After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value has been taken into account.

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: PLLRCLK  
 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 SYSCLK as follows:

000: HSISYS  
 001: HSE  
 010: PLLRCLK  
 011: LSI  
 100: LSE  
 Others: Reserved

The setting is forced by hardware to 000 (HSISYS selected):

- when the device exits Stop, Standby, or Shutdown mode, or
- when the HSE oscillator failure is detected while the HSE is selected directly or indirectly (PLLRCLK selected for SYSCLK and HSE selected as PLL input) as system clock source.

The setting is forced by hardware to 011 (LSI selected) when the LSE oscillator failure is detected while the LSE is selected as system clock source.

#### 5.4.4 PLL configuration register (RCC\_PLLCFGR)

Address offset: 0x0C

Reset value: 0x0000 1000

This register configures the PLL clock outputs according to the formulas:

- $f_{VCO} = f_{PLLIN} \times (N / M)$

- $f_{PLL P} = f_{VCO} / P$
- $f_{PLL Q} = f_{VCO} / Q$
- $f_{PLL R} = f_{VCO} / R$

where  $f_{PLL IN}$  is the PLL input clock frequency,  $f_{VCO}$  is the PLL VCO frequency, and P, Q and R are  $f_{VCO}$  division factors and  $f_{PLL P}$ ,  $f_{PLL Q}$  and  $f_{PLL R}$  the clock frequencies of the PLLPCLK, PLLQCLK and PLLRCLK PLL clock outputs, respectively.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL R[2:0]			PLL REN	PLL Q[2:0]			PLL QEN	Res.	Res.	PLL P[4:0]				PLL PEN	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PLL N[6:0]						Res.	PLL M[2:0]			Res.	Res.	PLL SRC[1:0]		
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

**Bits 31:29 PLLR[2:0]:** PLL VCO division factor R for PLLRCLK clock output

This bitfield is controlled by software. It sets the PLL VCO division factor R as follows:

- 000: Reserved
- 001: 2
- 010: 3
- 011: 4
- 100: 5
- 101: 6
- 110: 7
- 111: 8

The bitfield can be written only when the PLL is disabled.

The PLLRCLK clock can be selected as system clock.

**Caution:** The software must set this bitfield so as not to exceed 64 MHz on this clock.

**Bit 28 PLLREN:** PLLRCLK clock output enable

This bit is controlled by software to enable/disable the PLLRCLK clock output of the PLL:

- 0: Disable
- 1: Enable

This bit cannot be written when PLLRCLK output of the PLL is selected for system clock.

Disabling the PLLRCLK clock output, when not used, allows saving power.

**Bits 27:25 PLLQ[2:0]:** PLL VCO division factor Q for PLLQCLK clock output

This bitfield is controlled by software. It sets the PLL VCO division factor Q as follows:

- 000: Reserved
- 001: 2
- 010: 3
- 011: 4
- 100: 5
- 101: 6
- 110: 7
- 111: 8

The bitfield can be written only when the PLL is disabled.

*Note:* Only significant on devices integrating PLLQCLK, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

**Caution:** The software must set this bitfield so as not to exceed 128 MHz on this clock.



Bit 24 **PLLQEN**: PLLQCLK clock output enable

This bit is controlled by software to enable/disable the PLLQCLK clock output of the PLL:

0: Disable

1: Enable

Disabling the PLLQCLK clock output, when not used, allows saving power.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **PLL[4:0]P**: PLL VCO division factor P for PLLPCLK clock output

This bitfield is controlled by software. It sets the PLL VCO division factor P as follows:

00000: Reserved

00001: 2

...

11111: 32

The bitfield can be written only when the PLL is disabled.

**Caution:** The software must set this bitfield so as not to exceed 122 MHz on this clock.

Bit 16 **PLL[4:0]PEN**: PLLPCLK clock output enable

This bit is controlled by software to enable/disable the PLLPCLK clock output of the PLL:

0: Disable

1: Enable

Disabling the PLLPCLK clock output, when not used, allows saving power.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PLL[6:0]N**: PLL frequency multiplication factor N

This bit is controlled by software to set the division factor of the  $f_{VCO}$  feedback divider (that determines the PLL multiplication ratio) as follows:

0000000: Invalid

0000001: Reserved

...

0000111: Reserved

0001000: 8

0001001: 9

...

1010101: 85

1010110: 86

1010111: Reserved

...

1111111: Reserved

The bitfield can be written only when the PLL is disabled.

**Caution:** The software must set these bits so that the VCO output frequency is between 96 and 344 MHz.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PLLM**: Division factor M of the PLL input clock divider

This bit is controlled by software to divide the PLL input clock before the actual phase-locked loop, as follows:

- 000: 1
- 001: 2
- 010: 3
- 011: 4
- 100: 5
- 101: 6
- 110: 7
- 111: 8

The bitfield can be written only when the PLL is disabled.

**Caution:** The software must set these bits so that the PLL input frequency after the /M divider is between 2.66 and 16 MHz.

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

Bits 1:0 **PLLSRC**: PLL input clock source

This bit is controlled by software to select PLL clock source, as follows:

- 00: No clock
- 01: Reserved
- 10: HSI16
- 11: HSE

The bitfield can be written only when the PLL is disabled.

When the PLL is not used, selecting 00 allows saving power.

### 5.4.5 RCC clock recovery RC register (RCC\_CRRCR)

This register applies to STM32G0B1xx and STM32G0C1xx only. It is reserved otherwise.

Address offset: 0x14

Reset value: 0b0000 0000 0000 0000 0000 000X XXXX XXXX

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.	HSI48CAL[8:0]								
							r	r	r	r	r	r	r	r	r

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

Bits 8:0 **HSI48CAL[8:0]**: HSI48 clock calibration

These bits are initialized at startup with the factory-programmed HSI48 calibration trim value.

### 5.4.6 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.	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.	PLL RDYIE	HSE RDYIE	HSI RDYIE	HSI48 RDYIE	LSE RDYIE	LSI RDYIE
										rw	rw	rw	rw	rw	rw

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

Bit 5 **PLLRDYIE**: PLL ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLL lock:

0: Disable

1: Enable

Bit 4 **HSERDYIE**: HSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization:

0: Disable

1: Enable

Bit 3 **HSIRDYIE**: HSI16 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization:

0: Disable

1: Enable

Bit 2 **HSI48RDYIE**: HSI48 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI48 oscillator stabilization:

0: Disable

1: Enable

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

### 5.4.7 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.	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.	PLL RDYF	HSE RDYF	HSI RDYF	HSI48 RDYF	LSE RDYF	LSI RDYF
						r	r			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  
 Set by hardware when a failure is detected in the LSE oscillator.  
 Cleared by software by setting the LSECSSC bit.  
 0: No clock security interrupt caused by LSE clock failure  
 1: Clock security interrupt caused by LSE clock failure

**Bit 8 CSSF:** HSE clock security system interrupt flag  
 Set by hardware when a failure is detected in the HSE oscillator.  
 Cleared by software setting the CSSC bit.  
 0: No clock security interrupt caused by HSE clock failure  
 1: Clock security interrupt caused by HSE clock failure

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

**Bit 5 PLLRDYF:** PLL ready interrupt flag  
 Set by hardware when the PLL locks and PLLRDYIE is set.  
 Cleared by software setting the PLLRDYC bit.  
 0: No clock ready interrupt caused by PLL lock  
 1: Clock ready interrupt caused by PLL lock

**Bit 4 HSERDYF:** HSE ready interrupt flag  
 Set by hardware when the HSE clock becomes stable and HSERDYIE is set.  
 Cleared by software setting the HSERDYC bit.  
 0: No clock ready interrupt caused by the HSE oscillator  
 1: Clock ready interrupt caused by the HSE oscillator

**Bit 3 HSIRDYF:** HSI16 ready interrupt flag  
 Set by hardware when the HSI16 clock becomes stable and HSIRDYIE is set in a response to setting the HSION (refer to [Clock control register \(RCC\\_CR\)](#)). When HSION is not set but the HSI16 oscillator is enabled by the peripheral through a clock request, this bit is not set and no interrupt is generated.  
 Cleared by software setting the HSIRDYC bit.  
 0: No clock ready interrupt caused by the HSI16 oscillator  
 1: Clock ready interrupt caused by the HSI16 oscillator

**Bit 2 HSI48RDYF:** HSI48 ready interrupt flag  
 Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set in a response to setting the HSI48ON (refer to [Clock control register \(RCC\\_CR\)](#)). When HSI48ON 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 HSI48RDYC bit.  
 0: No clock ready interrupt caused by the HSI48 oscillator  
 1: Clock ready interrupt caused by the HSI48 oscillator

- Bit 1 **LSERDYF**: LSE ready interrupt flag  
 Set by hardware when the LSE clock becomes stable and LSERDYIE is set.  
 Cleared by software setting the LSERDYC bit.  
 0: No clock ready interrupt caused by the LSE oscillator  
 1: Clock ready interrupt caused by the LSE oscillator
- Bit 0 **LSIRDYF**: LSI ready interrupt flag  
 Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.  
 Cleared by software setting the LSIRDYC bit.  
 0: No clock ready interrupt caused by the LSI oscillator  
 1: Clock ready interrupt caused by the LSI oscillator

### 5.4.8 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.	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.	PLL RDYC	HSE RDYC	HSI RDYC	HSI48 RDYC	LSE RDYC	LSI RDYC
						w	w			w	w	w	w	w	w

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 7:6 Reserved, must be kept at reset value.
- Bit 5 **PLLRDYC**: PLL ready interrupt clear  
 This bit is set by software to clear the PLLRDYF flag.  
 0: No effect  
 1: Clear PLLRDYF flag
- Bit 4 **HSERDYC**: HSE ready interrupt clear  
 This bit is set by software to clear the HSERDYF flag.  
 0: No effect  
 1: Clear HSERDYF flag
- Bit 3 **HSIRDYC**: HSI16 ready interrupt clear  
 This bit is set software to clear the HSIRDYF flag.  
 0: No effect  
 1: Clear HSIRDYF flag

- Bit 2 **HSI48RDYC**: HSI48 ready interrupt clear  
 This bit is set software to clear the HSI48RDYF flag.  
 0: No effect  
 1: Clear HSI48RDYF flag
- 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

### 5.4.9 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.	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.	GPIOF RST	GPIOE RST <sup>(1)</sup>	GIPOD RST	GPIOC RST	GPIOB RST	GPIOA RST
										rW	rW	rW	rW	rW	rW

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

- 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 **GPIOERST**: I/O port E reset<sup>(1)</sup>  
 This bit is set and cleared by software.  
 0: no effect  
 1: Reset I/O port E
- Bit 3 **GIPODRST**: 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

### 5.4.10 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.	RNG RST	Res.	AES RST
													rw		rw
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.	Res.	Res.	Res.	Res.	Res.	DMA2 RST	DMA1 RST
			rw				rw							rw	rw

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

- Bit 18 **RNGRST**: Random number generator reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset RNG

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 17 Reserved, must be kept at reset value.

- Bit 16 **AESRST**: AES hardware accelerator reset<sup>(1)</sup>  
 Set and cleared by software.  
 0: No effect  
 1: Reset AES

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bits 15:13 Reserved, must be kept at reset value.

- Bit 12 **CRCRST**: CRC reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset CRC

Bits 11:9 Reserved, must be kept at reset value.



Bit 8 **FLASHRST**: Flash memory interface reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset flash memory interface  
 This bit can only be set when the flash memory is in power down mode.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DMA2RST**: DMA2 and DMAMUX reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset DMA2 and DMAMUX

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 0 **DMA1RST**: DMA1 and DMAMUX reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset DMA1 and DMAMUX

### 5.4.11 APB peripheral reset register 1 (RCC\_APBSTR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 RST	LPTIM2 RST	DAC1 RST	PWR RST	DBG RST	UCPD2 RST	UCPD1 RST	CEC RST	I2C3 RST	I2C2 RST	I2C1 RST	LP UART1 RST	USART4 RST	USART3 RST	USART2 RST	CRSR ST
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
SPI3 RST	SPI2 RST	USB RST	FDCA N RST	Res.	Res.	USART6 RST	USART5 RST	LP UART2 RST	Res.	TIM7 RST	TIM6 RST	Res.	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw	rw	rw			rw	rw	rw		rw	rw		rw	rw	rw

Bit 31 **LPTIM1RST**: Low Power Timer 1 reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset LPTIM1

Bit 30 **LPTIM2RST**: Low Power Timer 2 reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset LPTIM2

Bit 29 **DAC1RST**: DAC1 interface reset<sup>(1)</sup>  
 Set and cleared by software.  
 0: No effect  
 1: Reset DAC1 interface

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



Bit 27 **DBG\_RST**: Debug support reset  
Set and cleared by software.  
0: No effect  
1: Reset DBG

Bit 26 **UCPD2\_RST**: UCPD2 reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset UCPD2

Bit 25 **UCPD1\_RST**: UCPD1 reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset UCPD1

Bit 24 **CEC\_RST**: HDMI CEC reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset the HDMI CEC

Bit 23 **I2C3\_RST**: I2C3 reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset I2C3

Bit 22 **I2C2\_RST**: I2C2 reset  
Set and cleared by software.  
0: No effect  
1: Reset I2C2

Bit 21 **I2C1\_RST**: I2C1 reset  
Set and cleared by software.  
0: No effect  
1: Reset I2C1

Bit 20 **LPUART1\_RST**: LPUART1 reset  
Set and cleared by software.  
0: No effect  
1: Reset LPUART1

Bit 19 **USART4\_RST**: USART4 reset  
Set and cleared by software.  
0: No effect  
1: Reset USART4

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 18 **USART3\_RST**: USART3 reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset USART3

*Note:*

- Bit 17 **USART2RST**: USART2 reset  
Set and cleared by software.  
0: No effect  
1: Reset USART2
- Bit 16 **CRSRST**: CRS reset  
Set and cleared by software.  
0: No effect  
1: Reset CRS  
*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*
- Bit 15 **SPI3RST**: SPI3 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI3
- Bit 14 **SPI2RST**: SPI2 reset  
Set and cleared by software.  
0: No effect  
1: Reset SPI2
- Bit 13 **USBRST**: USB reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset USB
- Bit 12 **FDCANRST**: FDCAN reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset FDCAN
- Bits 11:10 Reserved, must be kept at reset value.
- Bit 9 **USART6RST**: USART3 reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset USART6
- Bit 8 **USART5RST**: USART3 reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset USART5
- Bit 7 **LPUART2RST**: LPUART2 reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset LPUART2
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **TIM7RST**: TIM7 timer reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset TIM7

- Bit 4 **TIM6RST**: TIM6 timer reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset TIM6
- Bit 3 Reserved, must be kept at reset value.
- Bit 2 **TIM4RST**: TIM3 timer reset<sup>(1)</sup>  
Set and cleared by software.  
0: No effect  
1: Reset TIM4
- Bit 1 **TIM3RST**: TIM3 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM3
- Bit 0 **TIM2RST**: TIM2 timer reset  
Set and cleared by software.  
0: No effect  
1: Reset TIM2

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 5.4.12 APB peripheral reset register 2 (RCC\_APBSTR2)

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	TIM15 RST
											rw		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.	Res.	Res.	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 **TIM15RST**: TIM15 timer reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset TIM15 timer

Bit 15 **TIM14RST**: TIM14 timer reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset TIM14 timer

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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, COMP and VREFBUF reset  
 Set and cleared by software.  
 0: No effect  
 1: Reset SYSCFG + COMP + VREFBUF

### 5.4.13 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.	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.	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
										rw	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 **GPIOEEN**: I/O port E clock enable  
 This bit is set and cleared by software.  
 0: Disable  
 1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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

### 5.4.14 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.	RNG EN	Res.	AES EN
													rw		rw
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.	Res.	Res.	Res.	Res.	Res.	DMA2 EN	DMA1 EN
			rw				rw							rw	rw

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

Bit 18 **RNGEN**: Random number generator clock enable

Set and cleared by software.

0: Disable

1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 17 Reserved, must be kept at reset value.

Bit 16 **AESEN**: AES hardware accelerator

Set and cleared by software.

0: Disable

1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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 7:2 Reserved, must be kept at reset value.

Bit 1 **DMA2EN**: DMA2 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.

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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.

#### 5.4.15 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
LPTIM1 EN	LPTIM2 EN	DAC1 EN	PWR EN	DBG EN	UCPD2 EN	UCPD1 EN	CEC EN	I2C3 EN	I2C2 EN	I2C1 EN	LP UART1 EN	USART4 EN	USART3 EN	USART2 EN	CRSE N
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
SPI3 EN	SPI2 EN	USB EN	FDCA N EN	WWDG EN	RTC APB EN	USART 6 EN	USART 5 EN	LP UART2 EN	Res.	TIM7 EN	TIM6 EN	Res.	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw

- Bit 31 LPTIM1EN:** LPTIM1 clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 30 LPTIM2EN:** LPTIM2 clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 29 DAC1EN:** DAC1 interface clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 28 PWREN:** Power interface clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 27 DBGEN:** Debug support clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 26 UCPD2EN:** UCPD2 clock enable<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 25 UCPD1EN:** UCPD1 clock enable<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 24 CECEN:** HDMI CEC clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 23 I2C3EN:** I2C3 clock enable<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable

- Bit 22 **I2C2EN**: I2C2 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 21 **I2C1EN**: I2C1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 20 **LPUART1EN**: LPUART1 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 19 **USART4EN**: USART4 clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 18 **USART3EN**: USART3 clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 17 **USART2EN**: USART2 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 16 **CRSEN**: CRS clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 15 **SPI3EN**: SPI3 clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 14 **SPI2EN**: SPI2 clock enable  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 13 **USBEN**: USB clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 12 **FDCANEN**: FDCAN clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable



- 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
- Bit 9 **USART6EN**: USART6 clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 8 **USART5EN**: USART5 clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 7 **LPUART2EN**: LPUART2 clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **TIM7EN**: TIM7 timer clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 4 **TIM6EN**: TIM6 timer clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 3 Reserved, must be kept at reset value.
- Bit 2 **TIM4EN**: TIM4 timer clock enable<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 1 **TIM3EN**: TIM3 timer clock enable  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 0 **TIM2EN**: TIM2 timer clock enable  
Set and cleared by software.  
0: Disable  
1: Enable

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 5.4.16 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	TIM15 EN
											rw		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.	Res.	Res.	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 **TIM15EN**: TIM15 timer clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 15 **TIM14EN**: TIM14 timer clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable

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

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable

Bit 11 **TIM1EN**: TIM1 timer clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGEN**: SYSCFG, COMP and VREFBUF clock enable  
 Set and cleared by software.  
 0: Disable  
 1: Enable

### 5.4.17 I/O port in Sleep mode clock enable register (RCC\_IOPSMENR)

Address offset: 0x44

Reset value: 0x0000 003F

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.	GPIOF SMEN	GPIOE SMEN	GPIOD SMEN	GPIOC SMEN	GPIOB SMEN	GPIOA SMEN
										rw	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 **GPIOESMEN**: I/O port E clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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

### 5.4.18 AHB peripheral clock enable in Sleep/Stop mode register (RCC\_AHBSMENR)

Address offset: 0x48

Reset value: 0x0005 1303

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.	RNG SMEN	Res.	AES SMEN
													rw		rw
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.	Res.	Res.	Res.	Res.	Res.	DMA2 SMEN	DMA1 SMEN
			rw			rw	rw							rw	rw

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

Bit 18 **RNGSMEN**: RNG clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 17 Reserved, must be kept at reset value.

Bit 16 **AESSMEN**: AES hardware accelerator clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **SRAMSMEN**: SRAM clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bit 8 **FLASHSMEN**: Flash memory interface clock enable during Sleep mode

Set and cleared by software.

0: Disable

1: Enable

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **DMA2SMEN**: DMA2 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.

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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.

#### 5.4.19 APB peripheral clock enable in Sleep/Stop mode register 1 (RCC\_APBSMENR1)

Address offset: 0x4C

Reset value: 0xFFFF FFB7

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
LPTIM1SMEN	LPTIM2SMEN	DAC1SMEN	PWRSMEN	DBGSMEN	UCPD2SMEN	UCPD1SMEN	CECSMEN	I2C3SMEN	I2C2SMEN	I2C1SMEN	LP UART1SMEN	USART4SMEN	USART3SMEN	USART2SMEN	CRSSMEN
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
SPI3SMEN	SPI2SMEN	USBSMEN	FDCA N SMEN	WWDGSMEN	RTC APB SMEN	USART6SMEN	USART5SMEN	LP UART2SMEN	Res.	TIM7SMEN	TIM6SMEN	Res.	TIM4SMEN	TIM3SMEN	TIM2SMEN
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw

- Bit 31 **LPTIM1SMEN**: Low Power Timer 1 clock enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 30 **LPTIM2SMEN**: Low Power Timer 2 clock enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 29 **DAC1SMEN**: DAC1 interface clock enable during Sleep and Stop modes<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 28 **PWRSMEN**: Power interface clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 27 **DBGSMEN**: Debug support clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 26 **UCPD2SMEN**: UCPD2 clock enable during Sleep mode<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 25 **UCPD1SMEN**: UCPD1 clock enable during Sleep mode<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 24 **CECSMEN**: HDMI CEC clock enable during Sleep and Stop modes<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable
- Bit 23 **I2C3SMEN**: I2C3 clock enable during Sleep mode<sup>(1)</sup>  
 Set and cleared by software.  
 0: Disable  
 1: Enable

- Bit 22 **I2C2SMEN**: I2C2 clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 21 **I2C1SMEN**: I2C1 clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 20 **LPUART1SMEN**: LPUART1 clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 19 **USART4SMEN**: USART4 clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 18 **USART3SMEN**: USART3 clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 17 **USART2SMEN**: USART2 clock enable during Sleep and Stop modes  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 16 **CRSSMEN**: CRS clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 15 **SPI3SMEN**: SPI3 clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 14 **SPI2SMEN**: SPI2 clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 13 **USBSMEN**: USB clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 12 **FDCANSMEN**: FDCAN clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable

- Bit 11 **WWDGSMEN**: WWDG clock enable during Sleep mode  
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
- Bit 9 **USART6SMEN**: USART6 clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 8 **USART5SMEN**: USART5 clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 7 **LPUART2SMEN**: LPUART2 clock enable during Sleep and Stop modes<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **TIM7SMEN**: TIM7 timer clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 4 **TIM6SMEN**: TIM6 timer clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 3 Reserved, must be kept at reset value.
- Bit 2 **TIM4SMEN**: TIM4 timer clock enable during Sleep mode<sup>(1)</sup>  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 1 **TIM3SMEN**: TIM3 timer clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable
- Bit 0 **TIM2SMEN**: TIM2 timer clock enable during Sleep mode  
Set and cleared by software.  
0: Disable  
1: Enable

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved with zero reset value. Refer to [Section 1.4: Availability of peripherals](#).



### 5.4.20 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	TIM15 SMEN
											rw		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.	Res.	Res.	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 **TIM15SMEN**: TIM15 timer clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 15 **TIM14SMEN**: TIM14 timer clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable

Bit 14 **USART1SMEN**: USART1 clock enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: Disable  
 1: Enable



Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN**: SPI1 clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable

Bit 11 **TIM1SMEN**: TIM1 timer clock enable during Sleep mode  
 Set and cleared by software.  
 0: Disable  
 1: Enable

Bits 10:1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGSMEN**: SYSCFG, COMP and VREFBUF clock enable during Sleep and Stop modes  
 Set and cleared by software.  
 0: Disable  
 1: Enable

### 5.4.21 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]		RNGDIV[1:0]		RNGSEL[1:0]		Res.	TIM15SEL	Res.	TIM1SEL	LPTIM2SEL[1:0]		LPTIM1SEL[1:0]		Res.	Res.
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
I2C2I2S1SEL[1:0]		I2C1SEL[1:0]		LPUART1SEL[1:0]		LPUART2SEL[1:0]		Res.	CECSEL	USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw		rw	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 clock source for ADC:  
 00: System clock  
 01: PLLPCLK  
 10: HSI16  
 11: Reserved

Bits 29:28 **RNGDIV[1:0]**: Division factor of RNG clock divider<sup>(1)</sup>  
 This bitfield is controlled by software to select the division factor as follows:  
 00: 1  
 01: 2  
 10: 4  
 11: 8

Bits 27:26 **RNGSEL[1:0]**: RNG clock source selection<sup>(1)</sup>  
 This bitfield is controlled by software to select the RNG clock as follows:  
 00: No clock  
 01: HSI16  
 10: SYSCLK  
 11: PLLQCLK

Bit 25 Reserved, must be kept at reset value.

- Bit 24 **TIM15SEL**: TIM15 clock source selection<sup>(1)</sup>  
This bit is set and cleared by software. It selects TIM15 clock source as follows:  
0: TIMPCLK  
1: PLLQCLK
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 **TIM1SEL**: TIM1 clock source selection  
This bit is set and cleared by software. It selects TIM1 clock source as follows:  
0: TIMPCLK  
1: PLLQCLK<sup>(1)</sup>
- Bits 21:20 **LPTIM2SEL[1:0]**: LPTIM2 clock source selection  
This bitfield is controlled by software to select LPTIM2 clock source as follows:  
00: PCLK  
01: LSI  
10: HSI16  
11: LSE
- Bits 19:18 **LPTIM1SEL[1:0]**: LPTIM1 clock source selection  
This bitfield is controlled by software to select LPTIM1 clock source as follows:  
00: PCLK  
01: LSI  
10: HSI16  
11: LSE
- Bits 17:16 Reserved, must be kept at reset value.
- Bits 15:14 **I2C2I2S1SEL[1:0]**: I2C2/I2S1 clock source selection  
This bitfield is controlled by software to select I2S1/I2C2 clock source as follows:  
00: PCLK/SYSCLK  
01: SYSCLK/PLLCLK  
10: HSI16/HSI16  
11: Reserved/I2S\_CKIN  
*Note: On the STM32G0B1xx and STM32G0C1xx, the bitfield selects the clock to the I2C2 peripheral. On the other devices, it selects the clock to the I2S1 peripheral.*
- 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: HSI16  
11: Reserved
- Bits 11:10 **LPUART1SEL[1:0]**: LPUART1 clock source selection  
This bitfield is controlled by software to select LPUART1 clock source as follows:  
00: PCLK  
01: SYSCLK  
10: HSI16  
11: LSE

Bits 9:8 **LPUART2SEL[1:0]**: LPUART2 clock source selection<sup>(1)</sup>

This bitfield is controlled by software to select LPUART2 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

Bit 7 Reserved, must be kept at reset value.

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

This bit is set and cleared by software. It selects the HDMI CEC clock source as follows:

- 0: HSI16 divided by 488
- 1: LSE

Bits 5:4 **USART3SEL[1:0]**: USART3 clock source selection<sup>(1)</sup>

This bitfield is controlled by software to select USART2 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

Bits 3:2 **USART2SEL[1:0]**: USART2 clock source selection<sup>(1)</sup>

This bitfield is controlled by software to select USART2 clock source as follows:

- 00: PCLK
- 01: SYSCLK
- 10: HSI16
- 11: LSE

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: HSI16
- 11: LSE

1. Only significant on devices integrating the corresponding peripheral supporting independent clock selection (or supporting the corresponding function), otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#) and [Section 33.4: USART implementation](#).

### 5.4.22 Peripherals independent clock configuration register 2 (RCC\_CCIPR2)

This register is only available on STM32G0B1xx and STM32G0C1xx. Reserved on the other devices.

Address Offset: 0x58

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.	USBSEL[1:0]		Res.	Res.	FDCANSEL[1:0]		Res.	Res.	Res.	Res.	I2S2SEL[1:0]		I2S1SEL[1:0]	
		rw	rw			rw	rw					rw	rw	rw	rw

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

Bits 13:12 **USBSSEL[1:0]**: USB clock source selection

This bitfield is controlled by software to select the USB clock as follows:

- 00: HSI48
- 01: HSE
- 10: PLLQCLK
- 11: Reserved

*Note: This bitfield is only significant on devices integrating the corresponding peripheral and clock, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **FDCANSEL[1:0]**: FDCAN clock source selection

This bitfield is controlled by software to select the FDCAN clock as follows:

- 00: PCLK
- 01: PLLQCLK
- 10: HSE
- 11: Reserved

*Note: This bitfield is only significant on devices integrating the corresponding peripheral and clock, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

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

Bits 3:2 **I2S2SEL[1:0]**: I2S2 clock source selection

This bitfield is controlled by software to select I2S2 clock source as follows:

- 00: SYSCLK
- 01: PLLPCLK
- 10: HSI16
- 11: External I2S clock selected as I2S2

Bits 1:0 **I2S1SEL[1:0]**: I2S1 clock source selection

This bitfield is controlled by software to select I2S1 clock source as follows:

- 00: SYSCLK
- 01: PLLPCLK
- 10: HSI16
- 11: External I2S clock selected as I2S1

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

Up to three wait states are inserted in case of successive accesses to this register. As this register is outside of the  $V_{CORE}$  domain, it is write-protected upon reset. The DBP bit of the [Power control register 1 \(PWR\\_CR1\)](#) must be set to allow their modification. Refer to [Section 4.1.2: Battery backup of RTC domain on page 115](#) for further information.

The register bits are only reset upon RTC domain reset (see [Section 5.1.3: RTC domain reset](#)), except the LSCOSEL, LSCOEN, and BDRST 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.	BDRST
						rw	rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		Res.	LSE CSSD	LSE CSSON	LSEDRV[1:0]		LSE BYP	LSE RDY	LSEON
rw						rw	rw		r	rw	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 **BDRST**: RTC domain software reset

Set and cleared by software to reset the RTC domain:

0: No effect

1: Reset

Bit 15 **RTCEN**: RTC clock enable

Set and cleared by software. The bit enables clock to RTC and 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 BDRST 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

**Caution:** When the LSECSSD bit is set, the LSE oscillator is held under reset and can't be restarted. To clear the LSECSSD bit, reset the RTC domain using BDRST bit.

Bit 5 **LSECSSON**: CSS on LSE enable

Set by software to enable the clock security system on LSE (32 kHz) oscillator as follows:

0: Disable

1: Enable

LSECSSON must be enabled after the LSE oscillator is enabled (LSEON bit enabled) and ready (LSERDY flag set by hardware), and after the RTCSEL bit is selected.

Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD =1). In that case the software **must** disable the LSECSSON bit.

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator drive capability

Set by software to select the LSE oscillator drive capability as follows:

00: low driving capability

01: medium-low driving capability

10: medium-high driving capability

11: high driving capability

Applicable when the LSE oscillator is in Xtal mode, as opposed to bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

Set and cleared by software to bypass the LSE oscillator.

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

### 5.4.24 Control/status register (RCC\_CSR)

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, Standby, or Shutdown mode entry.  
Cleared by setting the RMVF bit.  
0: No illegal mode reset occurred  
1: Illegal mode reset occurred  
This operates only if nRST\_STOP, nRST\_STDBY or nRST\_SHDW option bits are cleared.
- Bit 30 **WWDGRSTF**: Window watchdog reset flag  
Set by hardware when a window watchdog reset occurs.  
Cleared by setting the RMVF bit.  
0: No window watchdog reset occurred  
1: Window watchdog reset occurred
- Bit 29 **IWDGRSTF**: Independent window watchdog reset flag  
Set by hardware when an independent watchdog reset domain occurs.  
Cleared by setting the RMVF bit.  
0: No independent watchdog reset occurred  
1: Independent watchdog reset occurred
- Bit 28 **SFTRSTF**: Software reset flag  
Set by hardware when a software reset occurs.  
Cleared by setting the RMVF bit.  
0: No software reset occurred  
1: Software reset occurred
- Bit 27 **PWRRSTF**: BOR or POR/PDR flag  
Set by hardware when a BOR or POR/PDR occurs.  
Cleared by setting the RMVF bit.  
0: No BOR or POR occurred  
1: BOR or POR occurred
- Bit 26 **PINRSTF**: Pin or system reset flag  
Set by hardware when a reset from the PF2-NRST pin occurs or when a system reset is triggered by any other source. Cleared by setting the RMVF bit.  
0: No system reset occurred  
1: System reset from PF2-NRST pin or from other source 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

### 5.4.25 RCC register map

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

**Table 37. RCC register map and reset values**

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	PLLRDY	PLLON	HSI48RDY	HSI48ON	Res.	Res.	CSSON	HSEBYP	HSERDY	HSEON	Res.	Res.	Res.	HSIDIV[2:0]	Res.	HSIRDY	HSIKERON	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0	0	0	0			0	0	0	0				0	0	0	1	0	1							
0x04	RCC_ICSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[6:0]				HSICAL[7:0]									
	Reset value																			1	0	0	0	0	0	0	X	X	X	X	X	X	X
0x08	RCC_CFGR	MCO2PRE[3:0]			MCOSEL[3:0]			MCO2PRE[3:0]			MCO2SEL[3:0]			Res.	PPRE[2:0]			HPRE[3:0]			Res.	Res.	SWS[2:0]			SW[2:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x0C	RCC_PLLCFGR	PLL2R[2:0]		PLL2REN		PLL2Q[2:0]		PLL2QEN		Res.	Res.	PLL4P[4:0]			PLL2PEN		Res.	PLL2N[6:0]						Res.	PLL2M[2:0]		Res.	Res.	PLL2SRC[1:0]				
	Reset value	0	0	0	0	0	0	0	0			0	0	0	0	0	0		0	0	1	0	0	0	0		0	0	0			0	0
0x10	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.	Res.
0x14	RCC_CRRCR	Res.																							HSI48CAL[8:0]								
	Reset value																									x	x	x	x	x	x	x	x

Table 37. RCC register map and reset values (continued)

Off- set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x18	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSF	CSSF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																							0	0										
0x20	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSECSSC	CSSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																							0	0										
0x24	RCC_IOPRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	RCC_AHBRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGRST	Res.	AESRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value														0		0																		
0x2C	RCC_APBRRSTR1	LPTIM1RST	LPTIM2RST	DAC1RST	PWRRST	DBGRRST	UCPD2RST	UCPD1RST	CECRST	I2C3RST	I2C2RST	I2C1RST	LPUART1RST	USART4RST	USART3RST	USART2RST	CRSRST	SPI3RST	SPI2RST	USBRST	FDCANRST	Res.	Res.	USART6RST	USART5RST	LPUART2RST	Res.	TIM7RST	TIM6RST	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
0x30	RCC_APBRRSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x34	RCC_IOPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 37. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x38	RCC_AHBENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEN	Res.	AESEN	Res.	Res.	Res.	Res.	CRCE	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMA2EN	DMA1EN	
	Reset value														0		0					0			1							0	0		
0x3C	RCC_APBENR1	LPTIM1EN	LPTIM2EN	DAC1EN	PWREN	DBGEN	UCPD2EN	UCPD1EN	CECEN	I2C3EN	I2C2EN	I2C1EN	LPUART1EN	USART4EN	USART3EN	USART2EN	CRSEN	SPI3EN	SPI2EN	USBEN	FDCANEN	WWDGEN	RTCAPBEN	USART6EN	USART5EN	LPUART2EN	Res.	TIM7EN	TIM6EN	Res.	Res.	Res.	Res.	TIM3EN	TIM2EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x40	RCC_APBENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADCEN	Res.	TIM17EN	TIM16EN	TIM15EN	TIM14EN	USART11EN	Res.	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSCFGEN
	Reset value												0		0	0	0	0	0		0	0												0	0
0x44	RCC_IOPSMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																												1	1	1	1	1	1	1
0x48	RCC_AHBSMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGSMEN	Res.	AESSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														1		1																1	1	1
0x4C	RCC_APBSMENR1	LPTIM1SMEN	LPTIM2SMEN	DAC1SMEN	PWRSMEN	DBGSMEN	UCPD2SMEN	UCPD1SMEN	CECSMEN	I2C3SMEN	I2C2SMEN	I2C1SMEN	LPUART1SMEN	USART4SMEN	USART3SMEN	USART2SMEN	CRSSMEN	SPI3SMEN	SPI2SMEN	USBSMEN	FDCANSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1														
0x50	RCC_APBSMENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN	TIM14SMEN	USART1SMEN	Res.	SPI1SMEN	TIM1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														1	1	1	1	1	1		1	1												1

Table 37. RCC register map and reset values (continued)

Off-set	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
0x54	RCC_CCIPR	ADCSEL[1:0]		RNGDIV[1:0]		RNGSEL[1:0]		Res.	TIM15SEL		TIM1SEL	LPTIM2SEL[1:0]		LPTIM1SEL[1:0]				I2C2/I2S1SEL[1:0]		I2C1SEL[1:0]		LPUART1SEL[1:0]		LPUART2SEL[1:0]			CECSEL	USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[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
0x58	RCC_CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																			USBSEL[1:0]	0	0			FDCANSEL[1:0]	0							
0x5C	RCC_BDCR	Res.	Res.	Res.	Res.	Res.	Res.	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0	0								0	0							0	0							
0x60	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PWRRSTF	PINRSTF	OBLRSTF	Res.	RMVF	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																							

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 6 Clock recovery system (CRS)

This section applies only to STM32G0B1 and STM32G0C1 devices.

### 6.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from alternative synchronization or generated by the user software.

### 6.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity (see [Section 6.4.2: CRS internal signals](#))
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

### 6.3 CRS implementation

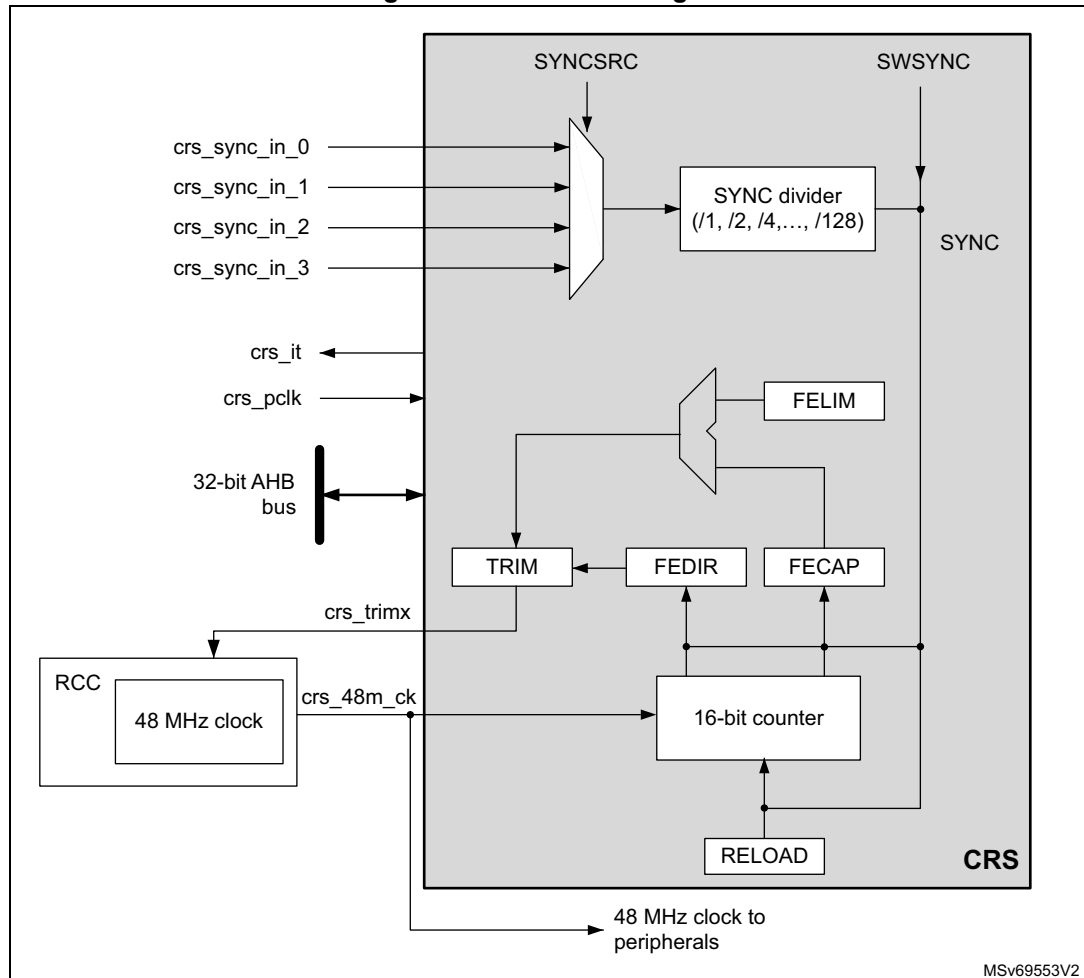
Table 38. CRS features

Feature	CRS
TRIM width	7 bits

## 6.4 CRS functional description

### 6.4.1 CRS block diagram

Figure 15. CRS block diagram



### 6.4.2 CRS internal signals

Below the list of CRS internal signals.

Table 39. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
crs_48m_ck	Digital input	HSI48 48 MHz clock
crs_trim[0:6]	Digital output	HSI48 oscillator smooth trimming value

Table 39. CRS internal input/output signals (continued)

Internal signal name	Signal type	Description
crs_sync_in_0	Digital input	SYNC signal source
crs_sync_in_1		
crs_sync_in_2		
crs_sync_in_3		

Table 40. CRS interconnection

Internal signal name	Description
crs_sync_in_0	GPIO selected as SYNC signal source
crs_sync_in_1	LSE selected as SYNC signal source
crs_sync_in_2	USB SOF selected as SYNC signal source (default)
crs_sync_in_3	Reserved

### 6.4.3 Synchronization input

The CRS synchronization source (crs\_sync\_in\_x) can be selected through SYNC\_SRC[1:0] bitfield of CRS\_CFGR register (refer to [Section 6.4.2: CRS internal signals](#) for the possible sources). For a better robustness of the crs\_sync\_in\_x input, a simple digital filter (2 out of 3 majority votes, sampled by the 48 MHz clock) is implemented to filter out glitches. In addition, the source signal has a configurable polarity (selected through SYNC\_POL bit of CRS\_CFGR). The signal can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to CRS\_CFGR register.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS\_CR register.

For more information on the CRS synchronization source configuration, refer to [CRS configuration register \(CRS\\_CFGR\)](#).

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS\_CR register.

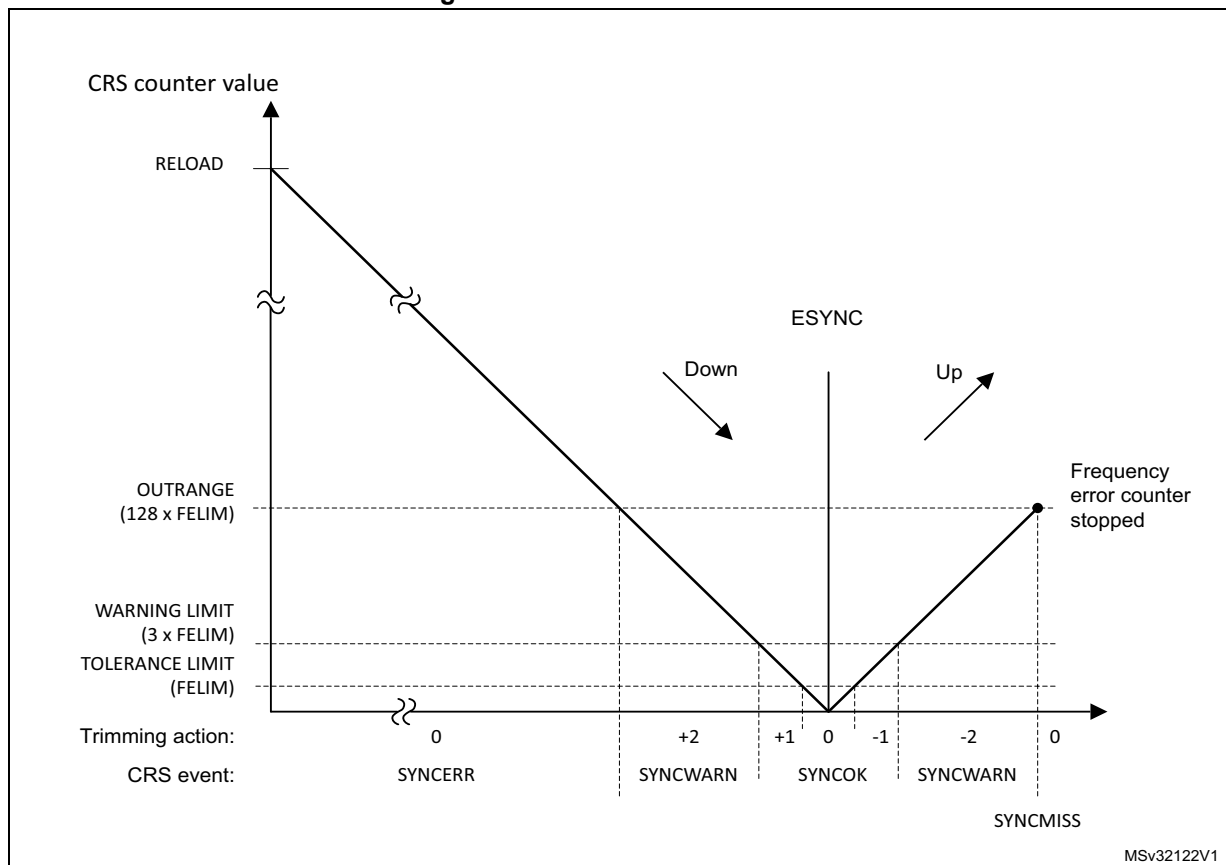
### 6.4.4 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OTRANGE limit is defined as the frequency error limit (FELIM field of the CRS\_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS\_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual

frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 16. CRS counter behavior



### 6.4.5 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS\_CFGR register
- WARNING LIMIT, defined as  $3 \times \text{FELIM}$  value
- OUTRANGE (error limit), defined as  $128 \times \text{FELIM}$  value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS\_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated



- TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

*Note:* If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

*In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS\_CR register), the TRIM field of CRS\_CR is adjusted by hardware and is read-only.*

## 6.4.6 CRS initialization and configuration

### RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$\text{RELOAD} = (f_{\text{TARGET}} / f_{\text{SYNC}}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

### FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$\text{FELIM} = (f_{\text{TARGET}} / f_{\text{SYNC}}) * \text{STEP}[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to  $(f_{\text{TARGET}} / f_{\text{SYNC}}) = 48000$ , and to a typical trimming step size of 0.14%.

*Note:* The trimming step size depends upon the product, check the datasheet for accurate setting.

**Caution:** There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than  $128 * \text{FELIM}$  value (OUTRANGE limit).

## 6.5 CRS in low-power modes

Table 41. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.
Shutdown	The peripheral is powered down and must be reinitialized after exiting Shutdown mode.

## 6.6 CRS interrupts

Table 42. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

## 6.7 CRS registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

### 6.7.1 CRS control register (CRS\_CR)

Address offset: 0x00

Reset value: 0x0000 4000

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.	TRIM[6:0]							SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNCE	ERRIE	SYNCWARNIE	SYNCOKIE
	rw	rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

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

Bits 14:8 **TRIM[6:0]**: HSI48 oscillator smooth trimming

The default value of the HSI48 oscillator smooth trimming is 64, which corresponds to the middle of the trimming interval.

- Bit 7 **SWSYNC**: Generate software SYNC event  
 This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.  
 0: No action  
 1: A software SYNC event is generated.
- Bit 6 **AUTOTRIMEN**: Automatic trimming enable  
 This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to [Section 6.4.5](#) for more details.  
 0: Automatic trimming disabled, TRIM bits can be adjusted by the user.  
 1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.
- Bit 5 **CEN**: Frequency error counter enable  
 This bit enables the oscillator clock for the frequency error counter.  
 0: Frequency error counter disabled  
 1: Frequency error counter enabled  
 When this bit is set, the CRS\_CFGR register is write-protected and cannot be modified.
- Bit 4 Reserved, must be kept at reset value.
- Bit 3 **ESYNCF**: Expected SYNC interrupt enable  
 0: Expected SYNC (ESYNCF) interrupt disabled  
 1: Expected SYNC (ESYNCF) interrupt enabled
- Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable  
 0: Synchronization or trimming error (ERRF) interrupt disabled  
 1: Synchronization or trimming error (ERRF) interrupt enabled
- Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable  
 0: SYNC warning (SYNCWARNF) interrupt disabled  
 1: SYNC warning (SYNCWARNF) interrupt enabled
- Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable  
 0: SYNC event OK (SYNCOKF) interrupt disabled  
 1: SYNC event OK (SYNCOKF) interrupt enabled

### 6.7.2 CRS configuration register (CRS\_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS\_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

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

Bit 31 **SYNCPOL**: SYNC polarity selection  
 This bit is set and cleared by software to select the input polarity for the SYNC signal source.  
 0: SYNC active on rising edge (default)  
 1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection  
 These bits are set and cleared by software to select the SYNC signal source (see [Section 6.4.2: CRS internal signals](#)).  
 00: crs\_sync\_in\_0 selected as SYNC signal source  
 01: crs\_sync\_in\_1 selected as SYNC signal source  
 10: crs\_sync\_in\_2 selected as SYNC signal source  
 11: crs\_sync\_in\_3 selected as SYNC signal source

*Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the 48 MHz clock on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.*

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider  
 These bits are set and cleared by software to control the division factor of the SYNC signal.  
 000: SYNC not divided (default)  
 001: SYNC divided by 2  
 010: SYNC divided by 4  
 011: SYNC divided by 8  
 100: SYNC divided by 16  
 101: SYNC divided by 32  
 110: SYNC divided by 64  
 111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit  
 FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS\_ISR register. Refer to [Section 6.4.5](#) for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value  
 RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to [Section 6.4.4](#) for more details about counter behavior.

### 6.7.3 CRS interrupt and status register (CRS\_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[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
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

- Bits 31:16 **FECAP[15:0]**: Frequency error capture  
FECAP is the frequency error counter value latched in the time of the last SYNC event.  
Refer to [Section 6.4.5](#) for more details about FECAP usage.
- Bit 15 **FEDIR**: Frequency error direction  
FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.  
0: Up-counting direction, the actual frequency is above the target  
1: Down-counting direction, the actual frequency is below the target
- Bits 14:11 Reserved, must be kept at reset value.
- Bit 10 **TRIMOVF**: Trimming overflow or underflow  
This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.  
0: No trimming error signaled  
1: Trimming error signaled
- Bit 9 **SYNCMISS**: SYNC missed  
This flag is set by hardware when the frequency error counter reaches value  $FELIM * 128$  and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.  
0: No SYNC missed error signaled  
1: SYNC missed error signaled
- Bit 8 **SYNCERR**: SYNC error  
This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to  $FELIM * 128$ . This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software by setting the ERRC bit in the CRS\_ICR register.  
0: No SYNC error signaled  
1: SYNC error signaled
- Bits 7:4 Reserved, must be kept at reset value.
- Bit 3 **ESYNCF**: Expected SYNC flag  
This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS\_CR register. It is cleared by software by setting the ESYNCC bit in the CRS\_ICR register.  
0: No expected SYNC signaled  
1: Expected SYNC signaled
- Bit 2 **ERRF**: Error flag  
This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS\_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS\_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.  
0: No synchronization or trimming error signaled  
1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to  $FELIM * 3$ , but smaller than  $FELIM * 128$ . This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS\_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS\_ICR register.

- 0: No SYNC warning signaled
- 1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than  $FELIM * 3$ . This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS\_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS\_ICR register.

- 0: No SYNC event OK signaled
- 1: SYNC event OK signaled

### 6.7.4 CRS interrupt flag clear register (CRS\_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 3 **ESYNCC**: Expected SYNC clear flag

Writing 1 to this bit clears the ESYNCF flag in the CRS\_ISR register.

Bit 2 **ERRC**: Error clear flag

Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS\_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag

Writing 1 to this bit clears the SYNCWARNF flag in the CRS\_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag

Writing 1 to this bit clears the SYNCOKF flag in the CRS\_ISR register.

6.7.5 CRS register map

Table 43. CRS 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	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[6:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF				
	Reset value	0x0000 4000																																		
0x04	CRS_CFGR	SYNCPOL	Res.	SYNC SRC [1:0]	Res.	SYNC DIV [2:0]	FELIM[7:0]							RELOAD[15:0]																						
	Reset value	0		1	0		0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1		
0x08	CRS_ISR	FECAP[15:0]																	FEDIR	Res.	Res.	Res.	Res.	Res.	TRIMOVF	SYNCF	SYNCF	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		
0x0C	CRS_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																			

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

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

### 7.1 Introduction

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

### 7.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx\_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Configurable state of each I/O in Standby and Shutdown mode: analog, pull-up/down
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx\_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx\_BSRR) for bitwise write access to GPIOx\_ODR
- Locking mechanism (GPIOx\_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers (at most 8 AFs possible per I/O)
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

### 7.3 GPIO functional description

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

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

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



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

Figure 17 shows the basic structures of a standard I/O port bit. Table 44 gives the possible port bit configurations.

Figure 17. Basic structure of an I/O port bit

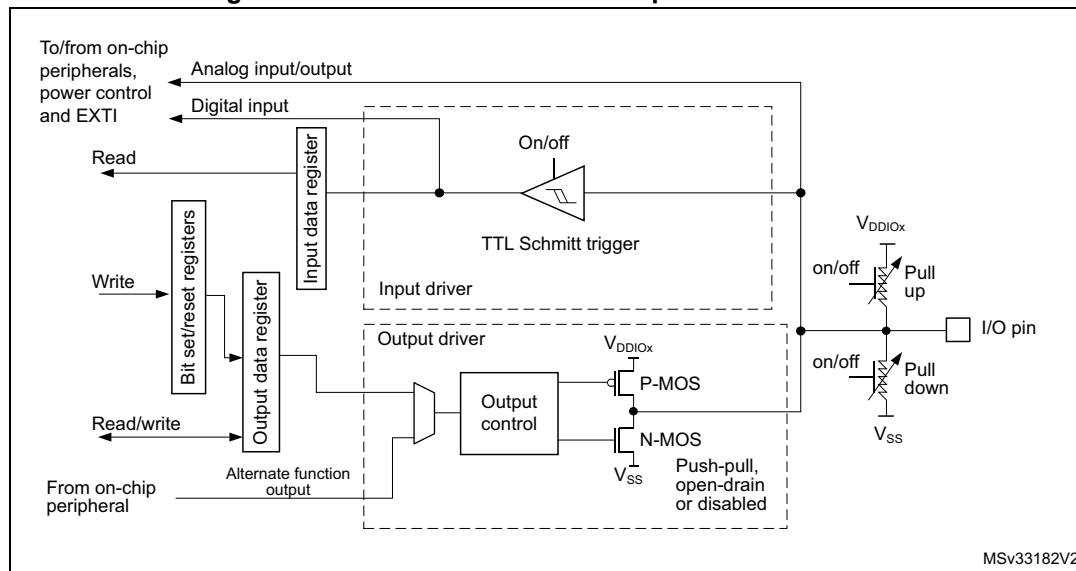


Table 44. Port bit configuration table<sup>(1)</sup>

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	1	Reserved	
	1		0	0	0	GP output	OD
	1		0	1	1	GP output	OD + PU
	1		1	0	0	GP output	OD + PD
	1		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	1	Reserved	
	1		0	0	0	AF	OD
	1		0	1	1	AF	OD + PU
	1		1	0	0	AF	OD + PD
	1		1	1	1	Reserved	

Table 44. 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		

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

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

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

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

- PA14: SWCLK in pull-down
- PA13: SWDIO in pull-up

*Note:* PA14 is shared with BOOT0 functionality. Caution is required as the debugging device can manipulate BOOT0 pin value.

Upon reset, the UCPD CCx lines present a pull-down resistor that can be disabled by setting the UCPDx\_STROBE bit of the SYSCFG\_CFGR1 register.

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

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

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

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

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

Each I/O pin has a multiplexer with up to eight alternate function inputs (AF0 to AF7) that can be configured through the GPIOx\_AFRL (for pin 0 to 7) and GPIOx\_AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx\_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

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

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

- **Debug function:** After each device reset these pins are assigned as alternate function pins immediately usable by the debugger host
- **GPIO:** Configure the desired I/O as output, input or analog in the GPIOx\_MODER register.
- **Alternate function:**
  - Connect the I/O to the desired AFx in one of the GPIOx\_AFRL or GPIOx\_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx\_OTYPER, GPIOx\_PUPDR and GPIOx\_OSPEEDER registers, respectively.
  - Configure the desired I/O as an alternate function in the GPIOx\_MODER register.

*Note: The Cortex-M0+ output EVENTOUT signal can be output as alternate function on several I/Os. An event can be signaled through the configured pin after executing SEV instruction.*

- **Additional functions:**
  - ADC, DAC, and COMP connection can be enabled in ADC, DAC, or COMP registers, regardless of the configured GPIO mode. When ADC, DAC or COMP uses a GPIO, it is recommended to configure the GPIO in analog mode, through the GPIOx\_MODER register.
  - For the additional functions such as RTC, TAMP, WKUPx and oscillators, configure the required function in the related RTC, TAMP, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

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

### 7.3.3 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.

### 7.3.4 I/O port state in low-power modes

In Standby and in Shutdown mode, GPIO digital interface is not powered and the configuration of GPIO ports is controlled by PWR instead. Use the PWR\_PUCRx and PWR\_PDCRx registers, together with the APC bit of the PWR\_CR3 register, to configure the GPIO pin states during the deep low-power modes, to prevent disturbing external components, buses, and to optimize power consumption.

*Note:* *Exiting Shutdown mode causes a power reset, which also resets the PWR registers, thus putting the GPIOs to analog mode until reinitialized by the application software.*

### 7.3.5 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 7.5.5: GPIO port input data register \(GPIOx\\_IDR\) \(x = A to F\)](#) and [Section 7.5.6: GPIO port output data register \(GPIOx\\_ODR\) \(x = A to F\)](#) for the register descriptions.

### 7.3.6 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.

### 7.3.7 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 7.5.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A to F\)](#)) 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 7.5.8: GPIO port configuration lock register \(GPIOx\\_LCKR\) \(x = A to F\)](#).

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

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

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

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

### 7.3.9 External interrupt/wake-up 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.

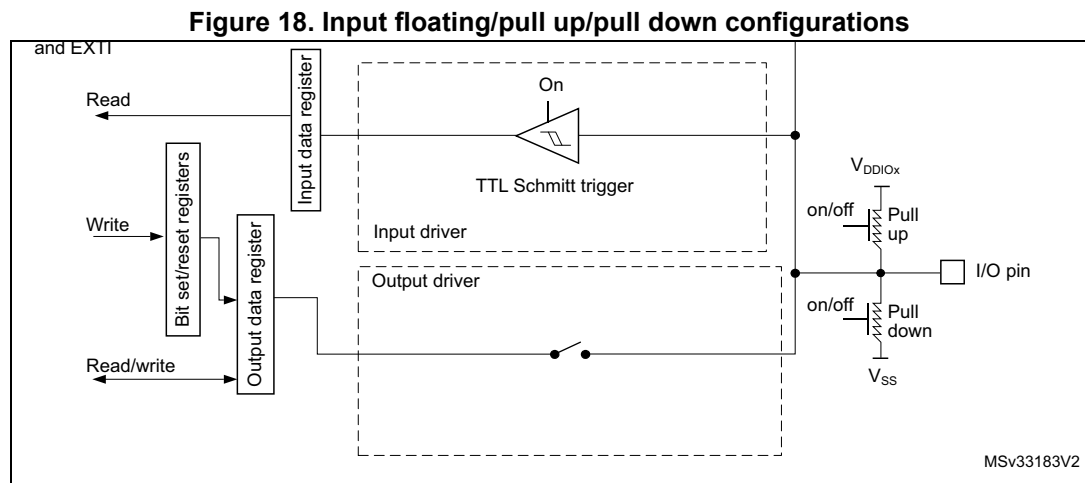
Refer to [Section 13: Extended interrupt and event controller \(EXTI\)](#).

### 7.3.10 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 18 shows the input configuration of the I/O port bit.



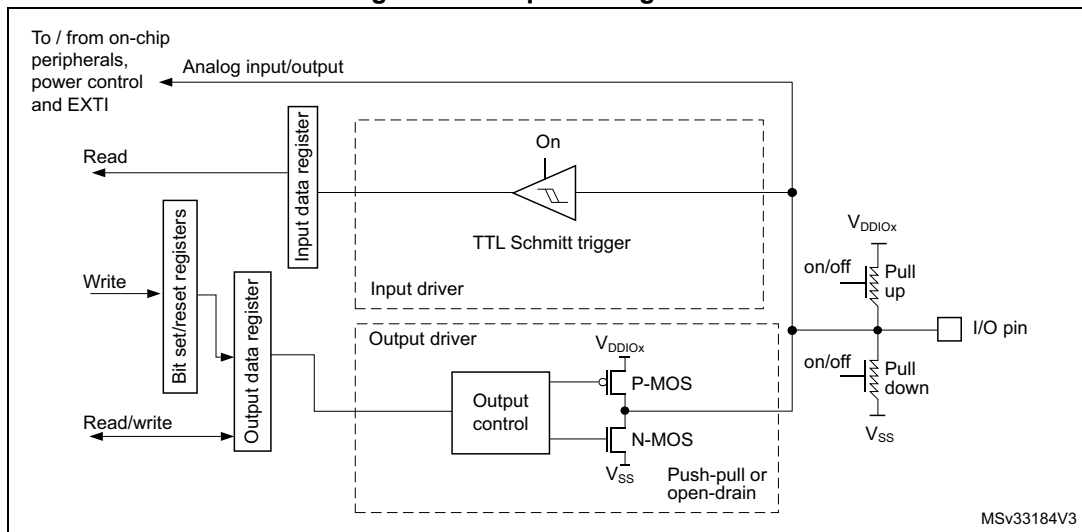
### 7.3.11 Output configuration

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 19 shows the output configuration of the I/O port bit.

Figure 19. Output configuration



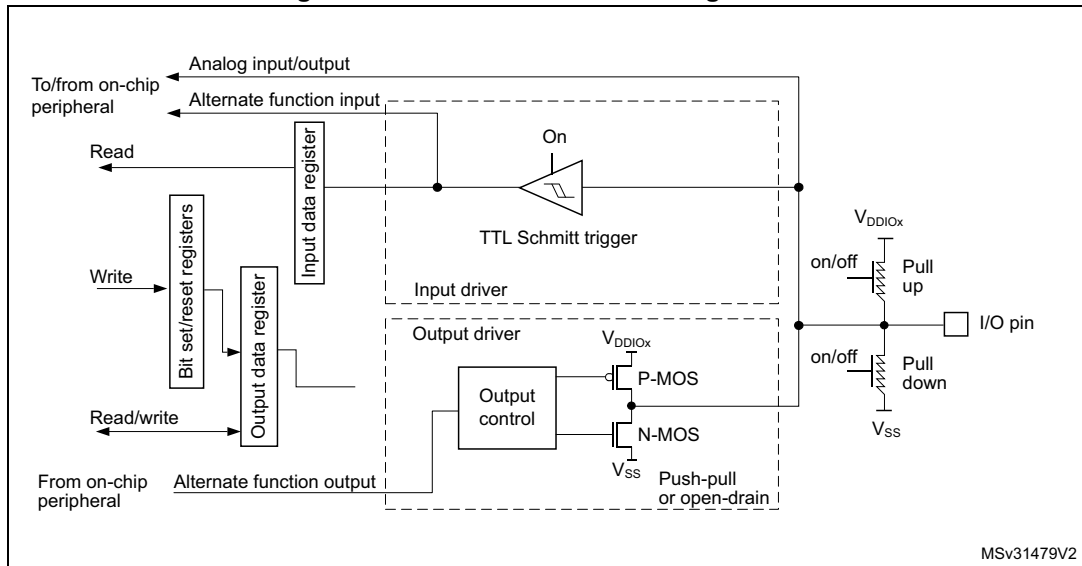
### 7.3.12 Alternate function configuration

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 20](#) shows the Alternate function configuration of the I/O port bit.

Figure 20. Alternate function configuration-



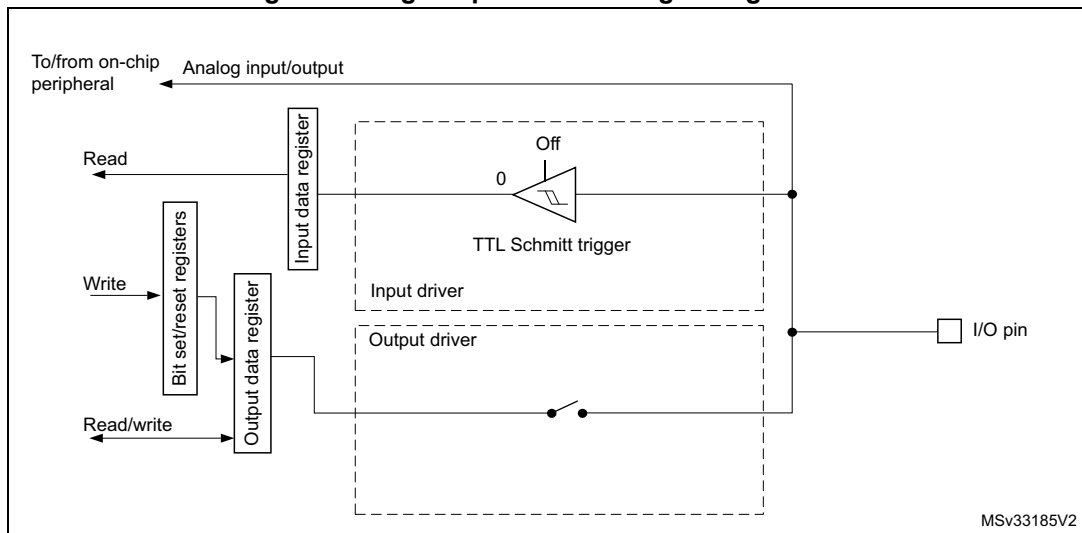
### 7.3.13 Analog configuration

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 21 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 21. High impedance-analog configuration





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

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

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

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

### 7.3.15 Using the GPIO pins in the RTC domain

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

For details about I/O control by the RTC, refer to [Section 30.3: RTC functional description](#).

### 7.3.16 USB PD / Dead battery support

In the absence of  $V_{DD}$  supply, the device using the Dead battery capability of the USB Type-C standard provides an internal pull-down resistor  $R_d$  on CC lines if the input level on DBCC pins is high. This is to signal VBUS supply request.

To enable this feature, it is necessary to connect UCPD\_DBCC1 to UCPD\_CC1 and UCPD\_DBCC2 to UCPD\_CC2. To disable the feature, it is necessary to connect UCPD\_DBCC1 and UCPD\_DBCC2 to ground. Refer to [Section 38: USB Type-C®/USB Power Delivery interface \(UCPD\)](#) for more detail.

*Note:* The DBCC pads (GPIOs of FT\_d type) present more leakage than standard GPIOs. Refer to product datasheet for values.

In applications that do not use the UCPD peripheral, disable the internal pull-down resistor  $R_d$  at startup through the strobe bits in SYSCFG registers.

In applications that use the UCPD peripheral, first configure the peripheral then load the configuration to the UCPD\_CCx GPIOs through the strobe bits in SYSCFG registers.

### 7.3.17 Reset pin (PF2-NRST) in GPIO mode

The PF2-NRST pin can be configured as reset I/O or as a GPIO.

To configure PF2-NRST as a GPIO (input, output, AF, or analog I/O), set the NRST\_MODE bitfield to GPIO mode in 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-NRST keeps acting as reset I/O.

The user must ensure that, upon power-on or exit from Standby or Shutdown mode, the level on the PF2-NRST pin can exceed the minimum  $V_{IH(NRST)}$  level specified in the device datasheet. Otherwise, the device does not exit the power-on reset. This applies to any PF2-NRST pin configuration set through the NRST\_MODE[1:0] bitfield, the GPIO mode inclusive.

When PF2-NRST 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.

## 7.4 GPIO in low-power modes

**Table 45. Effect of low-power modes on the GPIO**

Mode	Description
Sleep	No effect. GPIO (EXTI) interrupts cause the device to exit Sleep mode.
Stop	No effect. GPIO (EXTI) interrupts cause the device to exit Stop mode.
Standby	GPIO digital interface is powered down and must be reinitialized after exiting Standby mode. Wake-up pins can be configured to cause the device to exit Standby mode. GPIO's are set to analog mode by hardware. Pull-up or pull-down device can individually be enabled through the PWR_PUCRx and PWR_PDCRx registers, respectively, to keep the GPIOs at defined levels. Upon exiting Standby mode, the PWR_PUCRx and PWR_PDCRx register settings are effective until the APC bit of the PWR_CR3 register is cleared by the user software.
Shutdown	GPIO digital interface is powered down and must be reinitialized after exiting Shutdown mode. Wake-up pins can be configured to cause the device to exit Shutdown mode. GPIO's are set to analog mode by hardware. Pull-up or pull-down device can individually be enabled through the PWR_PUCRx and PWR_PDCRx registers, respectively, to keep the GPIOs at defined levels. Upon exiting Shutdown mode, the PWR_PUCRx and PWR_PDCRx registers as well as the GPIO ports are reset to their respective reset configuration.

## 7.5 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to [Table 46](#).

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

### 7.5.1 GPIO port mode register (GPIOx\_MODER) (x = A to F)

Address offset: 0x00

Reset value: 0xEBFF FFFF (port A)

Reset value: 0xFFFF FFFF (other ports)

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	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
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **MODEy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)  
 These bits are written by software to configure the I/O mode.  
 00: Input mode  
 01: General purpose output mode  
 10: Alternate function mode  
 11: Analog mode (reset state)

### 7.5.2 GPIO port output type register (GPIOx\_OTYPER) (x = A to 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
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:0 **OT[15:0]**: Port x configuration I/O pin y (y = 15 to 0)  
 These bits are written by software to configure the I/O output type.  
 0: Output push-pull (reset state)  
 1: Output open-drain

### 7.5.3 GPIO port output speed register (GPIOx\_OSPEEDR) (x = A to F)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 0000 (for other ports)

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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

00: Very low speed

01: Low speed

10: High speed

11: Very high speed

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

*Note: The FT\_c GPIOs cannot be set to high speed and to very high speed. Only very low speed and low speed settings are available, with specific characteristics.*

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

Address offset: 0x0C

Reset value: 0x2400 0000 (for port A)

Reset value: 0x0000 0000 (for other ports)

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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **PUPDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

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

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

### 7.5.5 GPIO port input data register (GPIOx\_IDR) (x = A to 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.	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
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 **ID[15:0]**: Port x input data I/O pin y (y = 15 to 0)

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

### 7.5.6 GPIO port output data register (GPIOx\_ODR) (x = A to F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:0 **OD[15:0]**: Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

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

### 7.5.7 GPIO port bit set/reset register (GPIOx\_BSRR) (x = A to 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 **BR[15:0]**: Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

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

Bits 15:0 **BS[15:0]**: Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

### 7.5.8 GPIO port configuration lock register (GPIOx\_LCKR) (x = A to 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
															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 16 **LCKK**: Lock key

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

0: Port configuration lock key not active

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

### 7.5.9 GPIO port alternate function low register (GPIOx\_AFRL) (x = A to 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]			
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELY[3:0]**: Alternate function selection for port x pin y (y = 7 to 0)  
 These bits are written by software to configure alternate function I/Os.  
 0000: AF0  
 0001: AF1  
 0010: AF2  
 0011: AF3  
 0100: AF4  
 0101: AF5  
 0110: AF6  
 0111: AF7  
 Other: Reserved

### 7.5.10 GPIO port alternate function high register (GPIOx\_AFRH) (x = A to 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	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
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELY[3:0]**: Alternate function selection for port x pin y (y = 15 to 8)  
 These bits are written by software to configure alternate function I/Os.  
 0000: AF0  
 0001: AF1  
 0010: AF2  
 0011: AF3  
 0100: AF4  
 0101: AF5  
 0110: AF6  
 0111: AF7  
 Other: Reserved

### 7.5.11 GPIO port bit reset register (GPIOx\_BRR) (x = A to 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.	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
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 **BR[15:0]**: Port x reset IO pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit

1: Reset the corresponding ODx bit



### 7.5.12 GPIO register map

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

**Table 46. GPIO register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	<b>GPIOx_MODER</b>	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[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]		
	Reset value port A	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
	Reset value port B to F	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
0x04	<b>GPIOx_OTYPER</b> (x = A to F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	<b>GPIOx_OSPEEDR</b> (x = A to F)	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[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]		
	Reset value port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Reset value port B to F	0	0	0	0	0	0	0	0	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>GPIOx_PUPDR</b> (x = A to F)	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[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]		
	Reset value port A	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Reset value port B to F	0	0	0	0	0	0	0	0	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>GPIOx_IDR</b> (x = A to F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x14	<b>GPIOx_ODR</b> (x = A to F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	<b>GPIOx_BSRR</b> (x = A to F)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	<b>GPIOx_LCKR</b> (x = A to F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	<b>GPIOx_AFRL</b> (x = A to F)	AFSEL7 [3:0]			AFSEL6 [3:0]			AFSEL5 [3:0]			AFSEL4 [3:0]			AFSEL3 [3:0]			AFSEL2 [3:0]			AFSEL1 [3:0]			AFSEL0 [3:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x24	<b>GPIOx_AFRH</b> (x = A to F)	AFSEL15 [3:0]			AFSEL14 [3:0]			AFSEL13 [3:0]			AFSEL12 [3:0]			AFSEL11 [3:0]			AFSEL10 [3:0]			AFSEL9 [3:0]			AFSEL8 [3:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x28	<b>GPIOx_BRR</b> (where x = A to F)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 8 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
- Enabling/disabling the analog switch booster
- Configuring the IR modulation signal and its output polarity
- Remapping of some I/O ports
- Remapping the memory located at the beginning of the code area
- Flag pending interrupts from each interrupt line
- Managing robustness feature

### 8.1 SYSCFG registers

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

This register is used for specific configurations of memory remap and to control special I/O features.

Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the hardware BOOT selection. After reset these bits take the value selected by the actual boot mode configuration.

Address offset: 0x00

Reset value: 0x0000 000X (X is the memory mode selected by the actual boot mode configuration)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3_FMP	I2C_PA10_FMP	I2C_PA9_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	UCPD2_STROBE	UCPD1_STROBE	BOOSTEN	IR_MOD [1:0]		IR_POL	PA12_RMP	PA11_RMP	Res.	MEM_MODE [1:0]	
					w	w	rw	rw	rw	rw	rw	rw		rw	rw

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

- Bit 24 I2C3\_FMP:** Fast Mode Plus (FM+) enable for I2C3  
This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on I/O ports configured as I2C3 through GPIOx\_AFR registers.  
0: Disable  
1: Enable  
The I2C FM+ driving capability on some I/O ports can also be enabled through their corresponding I2C\_xxx\_FMP bit. When I2C FM+ is enabled, the speed control is ignored.  
*Note: Only significant on devices integrating the corresponding peripheral or function, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*
- 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  
When this I/O port is configured for I2Cx, the I2C FM+ driving capability on this I/O port can also be enabled through the corresponding I2Cx\_FMP bit.
- 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  
When this I/O port is configured for I2Cx, the I2C FM+ driving capability on this I/O port can also be enabled through the corresponding I2Cx\_FMP bit.
- Bit 21 I2C2\_FMP:** Fast Mode Plus (FM+) enable for I2C2  
This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on I/O ports configured as I2C2 through GPIOx\_AFR registers.  
0: Disable  
1: Enable  
The I2C FM+ driving capability on some I/O ports can also be enabled through their corresponding I2C\_xxx\_FMP bit. When I2C FM+ is enabled, the speed control is ignored.
- Bit 20 I2C1\_FMP:** Fast Mode Plus (FM+) enable for I2C1  
This bit is set and cleared by software. It enables I<sup>2</sup>C FM+ driving capability on I/O ports configured as I2C1 through GPIOx\_AFR registers.  
0: Disable  
1: Enable  
The I2C FM+ driving capability on some I/O ports can also be enabled through their corresponding I2C\_xxx\_FMP bit. When I2C 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  
When this I/O port is configured for I2Cx, the I2C FM+ driving capability on this I/O port can also be enabled through the corresponding I2Cx\_FMP bit.
- 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  
When this I/O port is configured for I2Cx, the I2C FM+ driving capability on this I/O port can also be enabled through the corresponding I2Cx\_FMP bit.

- 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  
When this I/O port is configured for I2Cx, the I2C FM+ driving capability on this I/O port can also be enabled through the corresponding I2Cx\_FMP bit.
- 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  
When this I/O port is configured for I2Cx, the I2C FM+ driving capability on this I/O port can also be enabled through the corresponding I2Cx\_FMP bit.
- Bits 15:11 Reserved, must be kept at reset value.
- Bit 10 **UCPD2\_STROBE**: UCPD2 pull-down configuration strobe  
Upon power on, internal pull-down resistors on UCPD2 CC1 and CC2 pins are enabled (connected).  
The action of setting this bit has the following “strobing” effect:  
- when UCPD2 is disabled: disable UCPD pull-down resistors on CC1 and CC2  
- when UCPD2 is enabled, with CC1 and CC2 pin UCPD control bits configured: apply that configuration  
See [Section 38: USB Type-C®/USB Power Delivery interface \(UCPD\)](#) for details.  
*Note: Only significant on devices integrating the corresponding peripheral or function, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*
- Bit 9 **UCPD1\_STROBE**: UCPD1 pull-down configuration strobe  
Upon power on, internal pull-down resistors on UCPD1 CC1 and CC2 pins are enabled (connected).  
The action of setting this bit has the following “strobing” effect:  
- when UCPD1 is disabled: disable UCPD pull-down resistors on CC1 and CC2  
- when UCPD1 is enabled, with CC1 and CC2 pin UCPD control bits configured: apply that configuration  
See [Section 38: USB Type-C®/USB Power Delivery interface \(UCPD\)](#) for details.  
*Note: Only significant on devices integrating the corresponding peripheral or function, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*
- Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable  
This bit selects the way of supplying I/O analog switches:  
0: V<sub>DD</sub>  
1: Dedicated voltage booster (supplied by V<sub>DD</sub>)  
When using the analog inputs, setting to 0 is recommended for high V<sub>DD</sub>, setting to 1 for low V<sub>DD</sub> (less than 2.4 V).
- Bits 7:6 **IR\_MOD[1:0]**: IR Modulation Envelope signal selection  
This bitfield selects the signal for IR modulation envelope:  
00: TIM16  
01: USART1  
10: USART4 on STM32G071xx and STM32G081xx as well as STM32G0B1xx and STM32G0C1xx, USART2 on STM32G031xx and STM32G041xx as well as STM32G051xx and STM32G061xx  
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. In this case, the original PA10 pin (if available) is forced to analog mode.
- 0: No remap (PA12)
  - 1: Remap (PA10)

Bit 3 **PA11\_RMP**: PA11 pin remapping

- This bit is set and cleared by software. When set, it remaps the PA11 pin to operate as PA9 GPIO port, instead as PA11 GPIO port. In this case, the original PA9 pin (if available) is forced to analog mode.
- 0: No remap (PA11)
  - 1: Remap (PA9)

Bit 2 Reserved, must be kept at reset value.

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

- These bits are set and cleared by software. They control the memory internal mapping at address 0x0000 0000. After reset these bits take on the value selected by the actual boot mode configuration. Refer to [Section 2.5: Boot configuration](#) for more details.
- x0: Main flash memory mapped at 0x0000 0000
  - 01: System flash memory mapped at 0x0000 0000
  - 11: Embedded SRAM mapped at 0x0000 0000

### 8.1.2 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB2_CDEN	PB1_CDEN	PB0_CDEN	PA13_CDEN	PA6_CDEN	PA5_CDEN	PA3_CDEN	PA1_CDEN
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM_PEF	Res.	Res.	Res.	Res.	ECC_LOCK	PVD_LOCK	SRAM_PARITY_LOCK	LOCKUP_LOCK
							rc_w1					rw	rw	rw	rw

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

Bit 23 **PB2\_CDEN**: PB2 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PB2 pin.

0: Disable

1: Enable

Bit 22 **PB1\_CDEN**: PB1 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PB1 pin.

0: Disable

1: Enable

Bit 21 **PB0\_CDEN**: PB0 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PB0 pin.

0: Disable

1: Enable

Bit 20 **PA13\_CDEN**: PA13 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PA13 pin.

0: Disable

1: Enable

Bit 19 **PA6\_CDEN**: PA6 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PA6 pin.

0: Disable

1: Enable

Bit 18 **PA5\_CDEN**: PA5 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PA5 pin.

0: Disable

1: Enable

Bit 17 **PA3\_CDEN**: PA3 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PA3 pin.

0: Disable

1: Enable

Bit 16 **PA1\_CDEN**: PA1 clamping diode enable bit<sup>(1)</sup>

This bit is set and cleared by software. It enables (connects) a clamping diode to  $V_{DDIOx}$  on PA1 pin.

0: Disable

1: Enable

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

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

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

0: No SRAM parity error detected

1: SRAM parity error detected

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

**Bit 3 ECC\_LOCK:** ECC error lock bit

This bit is set by software and cleared by a system reset. It can be used to enable and lock the flash memory ECC 2-bit error detection signal connection to TIM1/15/16/17 Break input.  
 0: ECC error disconnected from TIM1/15/16/17 Break input  
 1: ECC error connected to TIM1/15/16/17 Break input

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

This bit is set by software and cleared by a system reset. It can be used to enable and lock the PVD connection to TIM1/15/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR\_CR register.  
 0: PVD interrupt disconnected from TIM1/15/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application.  
 1: PVD interrupt connected to TIM1/15/16/17 Break input, PVDE and PLS[2:0] bits are read only.

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

This bit is set by software and cleared by a system reset. It can be used to enable and lock the SRAM parity error signal connection to TIM1/15/16/17 Break input.  
 0: SRAM parity error disconnected from TIM1/15/16/17 Break input  
 1: SRAM parity error connected to TIM1/15/16/17 Break input

**Bit 0 LOCKUP\_LOCK:** Cortex<sup>®</sup>-M0+ LOCKUP bit enable bit

This bit is set by software and cleared by a system reset. It can be use to enable and lock the connection of Cortex<sup>®</sup>-M0+ LOCKUP (Hardfault) output to TIM1/15/16/17 Break input.  
 0: Cortex<sup>®</sup>-M0+ LOCKUP output disconnected from TIM1/15/16/17 Break input  
 1: Cortex<sup>®</sup>-M0+ LOCKUP output connected to TIM1/15/16/17 Break input

1. Only significant on devices integrating switchable clamping diodes, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 8.1.3 SYSCFG interrupt line 0 status register (SYSCFG\_ITLINE0)

A dedicated set of registers is implemented on the device to collect all pending interrupt sources associated with each interrupt line into a single register. This allows users to check by single read which peripheral requires service in case more than one source is associated to the interrupt line.

All bits in those registers are read only, set by hardware when there is corresponding interrupt request pending and cleared by resetting the interrupt source flags in the peripheral registers.

Address offset: 0x80

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG
															r

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

**Bit 0 WWDG:** Window watchdog interrupt pending flag



### 8.1.4 SYSCFG interrupt line 1 status register (SYSCFG\_ITLINE1)

Address offset: 0x84

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVMO UT	PVDO UT
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PVMOUT**: V<sub>DDIO2</sub> supply monitoring interrupt request pending (EXTI line 34).

Bit 0 **PVDOUT**: PVD supply monitoring interrupt request pending (EXTI line 16).

### 8.1.5 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.	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.	RTC	TAMP
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RTC**: RTC interrupt request pending (EXTI line 19)

*Note: Only significant on devices integrating VDDIO2 monitor, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 0 **TAMP**: Tamper interrupt request pending (EXTI line 21)

### 8.1.6 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.	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.	FLASH ITF	FLASH ECC
														r	r



Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FLASH\_ITF**: Flash memory interface interrupt request pending

Bit 0 **FLASH\_ECC**: Flash memory interface ECC interrupt request pending

### 8.1.7 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.	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.	CRS	RCC
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CRS**: CRS interrupt request pending

*Note: Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 0 **RCC**: Reset and clock control interrupt request pending

### 8.1.8 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.	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.	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

### 8.1.9 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.	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.	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

### 8.1.10 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

### 8.1.11 SYSCFG interrupt line 8 status register (SYSCFG\_ITLINE8)

Address offset: 0xA0

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB	UCPD2	UCPD1
													r	r	r

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

- Bit 2 **USB**: USB interrupt request pending<sup>(1)</sup>
- Bit 1 **UCPD2**: UCPD2 interrupt request pending (EXTI line 33)<sup>(1)</sup>
- Bit 0 **UCPD1**: UCPD1 interrupt request pending (EXTI line 32)<sup>(1)</sup>

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 8.1.12 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.	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.	DMA1_CH1
															r

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

Bit 0 **DMA1\_CH1**: DMA1 channel 1 interrupt request pending

### 8.1.13 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.	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.	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

### 8.1.14 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.	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.	DMA2_CH5	DMA2_CH4	DMA2_CH3	DMA2_CH2	DMA2_CH1	DMA1_CH7	DMA1_CH6	DMA1_CH5	DMA1_CH4	DMAM UX
						r	r	r	r	r	r	r	r	r	r

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

Bit 9 **DMA2\_CH5**: DMA2 channel 5 interrupt request pending<sup>(1)</sup>

Bit 8 **DMA2\_CH4**: DMA2 channel 4 interrupt request pending<sup>(1)</sup>

Bit 7 **DMA2\_CH3**: DMA2 channel 3 interrupt request pending<sup>(1)</sup>

Bit 6 **DMA2\_CH2**: DMA2 channel 2 interrupt request pending<sup>(1)</sup>

- Bit 5 **DMA2\_CH1**: DMA2 channel 1 interrupt request pending<sup>(1)</sup>
- Bit 4 **DMA1\_CH7**: DMA1 channel 7 interrupt request pending<sup>(1)</sup>
- Bit 3 **DMA1\_CH6**: DMA1 channel 6 interrupt request pending<sup>(1)</sup>
- Bit 2 **DMA1\_CH5**: DMA1 channel 5 interrupt request pending
- Bit 1 **DMA1\_CH4**: DMA1 channel 4 interrupt request pending
- Bit 0 **DMAMUX**: DMAMUX interrupt request pending

1. Only significant on devices integrating the corresponding DMA instance and channel, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 8.1.15 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.	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.	COMP 3	COMP 2	COMP 1	ADC
												r	r	r	r

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

- Bit 3 **COMP3**: Comparator 3 interrupt request pending (EXTI line 20)<sup>(1)</sup>
- Bit 2 **COMP2**: Comparator 2 interrupt request pending (EXTI line 18)<sup>(1)</sup>
- Bit 1 **COMP1**: Comparator 1 interrupt request pending (EXTI line 17)<sup>(1)</sup>
- Bit 0 **ADC**: ADC interrupt request pending

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 8.1.16 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.	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.	TIM1_B RK	TIM1_ UPD	TIM1_ TRG	TIM1_ CCU
												r	r	r	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

### 8.1.17 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.	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.	TIM1_CC
															r

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

Bit 0 **TIM1\_CC**: Timer 1 capture compare interrupt request pending

### 8.1.18 SYSCFG interrupt line 15 status register (SYSCFG\_ITLINE15)

Address offset: BCh

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2
															r

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

Bit 0 **TIM2**: Timer 2 interrupt request pending

### 8.1.19 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.	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.	TIM4	TIM3
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **TIM4**: Timer 4 interrupt request pending

*Note: Only significant on devices integrating TIM4, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 0 **TIM3**: Timer 3 interrupt request pending

### 8.1.20 SYSCFG interrupt line 17 status register (SYSCFG\_ITLINE17)

Address offset: 0xC4

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1	DAC	TIM6
													r	r	r

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

Bit 2 **LPTIM1**: Low-power timer 1 interrupt request pending (EXTI line 29)

Bit 1 **DAC**: DAC underrun interrupt request pending<sup>(1)</sup>

Bit 0 **TIM6**: Timer 6 interrupt request pending<sup>(1)</sup>

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 8.1.21 SYSCFG interrupt line 18 status register (SYSCFG\_ITLINE18)

Address offset: 0xC8

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2	TIM7
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LPTIM2**: Low-power timer 2 interrupt request pending (EXTI line 30)

Bit 0 **TIM7**: Timer 7 interrupt request pending

*Note: Only significant on devices integrating TIM7, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

### 8.1.22 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.	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.	TIM14
															r

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

Bit 0 **TIM14**: Timer 14 interrupt request pending

### 8.1.23 SYSCFG interrupt line 20 status register (SYSCFG\_ITLINE20)

Address offset: 0xD0

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM15
															r

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

Bit 0 **TIM15**: Timer 15 interrupt request pending

*Note: Only significant on devices integrating TIM15, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

### 8.1.24 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.	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.	FDCAN2_IT0	FDCAN1_IT0	TIM16
													r	r	r

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

Bit 2 **FDCAN2\_IT0**: FDCAN2 interrupt request pending

*Note: Only significant on devices integrating FDCAN2, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 1 **FDCAN1\_IT0**: FDCAN1 interrupt request pending

*Note: Only significant on devices integrating FDCAN1, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 0 **TIM16**: Timer 16 interrupt request pending

### 8.1.25 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.	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.	FDCAN2_IT1	FDCAN1_IT1	TIM17
													r	r	r

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

Bit 2 **FDCAN2\_IT1**: FDCAN2 interrupt request pending

*Note: Only significant on devices integrating FDCAN2, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 1 **FDCAN1\_IT1**: FDCAN1 interrupt request pending

*Note: Only significant on devices integrating FDCAN1, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

Bit 0 **TIM17**: Timer 17 interrupt request pending

### 8.1.26 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.	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.	I2C1
															r

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

Bit 0 **I2C1**: I2C1 interrupt request pending, combined with EXTI line 23

### 8.1.27 SYSCFG interrupt line 24 status register (SYSCFG\_ITLINE24)

Address offset: 0xE0

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3 <sup>(1)</sup>	I2C2
														r	r

1. Only significant on devices integrating I2C3, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **I2C3**: I2C3 interrupt request pending (EXTI line 22)<sup>(1)</sup>

Bit 0 **I2C2**: I2C2 interrupt request pending

### 8.1.28 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.	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.	SPI1
															r

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

Bit 0 **SPI1**: SPI1 interrupt request pending

### 8.1.29 SYSCFG interrupt line 26 status register (SYSCFG\_ITLINE26)

Address offset: 0xE8

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI3 <sup>(1)</sup>	SPI2
														r	r

1. Only significant on devices integrating SPI3, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SPI3**: SPI3 interrupt request pending<sup>(1)</sup>

Bit 0 **SPI2**: SPI2 interrupt request pending

### 8.1.30 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.	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.	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

### 8.1.31 SYSCFG interrupt line 28 status register (SYSCFG\_ITLINE28)

Address offset: 0xF0

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPUART2 <sup>(1)</sup>	USART 2
														r	r

1. Only significant on devices integrating LPUART2, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **LPUART2**: LPUART2 interrupt request pending<sup>(1)</sup>

Bit 0 **USART2**: USART2 interrupt request pending (EXTI line 26)

### 8.1.32 SYSCFG interrupt line 29 status register (SYSCFG\_ITLINE29)

Address offset: 0xF4

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART 6	USART 5	LPUAR T1	USART 4	USART 3
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **USART6**: USART6 interrupt request pending<sup>(1)</sup>

Bit 3 **USART5**: USART5 interrupt request pending<sup>(1)</sup>

Bit 2 **LPUART1**: LPUART1 interrupt request pending (EXTI line 28)

Bit 1 **USART4**: USART4 interrupt request pending<sup>(1)</sup>

Bit 0 **USART3**: USART3 interrupt request pending (EXTI line 28)<sup>(1)</sup>

1. Only significant on devices integrating the corresponding peripheral, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 8.1.33 SYSCFG interrupt line 30 status register (SYSCFG\_ITLINE30)

Address offset: 0xF8

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEC
															r

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

Bit 0 **CEC**: CEC interrupt request pending (EXTI line 27)

*Note: Only significant on devices integrating CEC, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).*

### 8.1.34 SYSCFG interrupt line 31 status register (SYSCFG\_ITLINE31)

Address offset: 0xFC

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AES	RNG
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **AES**: AES interrupt request pending

Note: Only significant on devices integrating AES, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

Bit 0 **RNG**: RNG interrupt request pending

Note: Only significant on devices integrating RNG, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

### 8.1.35 SYSCFG register map

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

Table 47. SYSCFG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	<b>SYSCFG_CFGR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C3_FMP	I2C_PA10_FMP	I2C_PA9_FMP	I2C2_FMP	I2C1_FMP	I2C_PB9_FMP	I2C_PB8_FMP	I2C_PB7_FMP	I2C_PB6_FMP	Res.	Res.	Res.	Res.	Res.	UCPD2_STROBE	UCPD2_STROBE	BOOSTEN	IR_MOD	IR_POL	PA12_RMP	PA11_RMP	Res.	MEM_MODE[1:0]			
	Reset value								0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0		X	X	
0x04 to 0x17	<b>Reserved</b>	Reserved																																
0x18	<b>SYSCFG_CFGR2</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.
	Reset value																									0					0	0	0	0
0x1D to 0x7F	<b>Reserved</b>	Reserved																																
0x80	<b>SYSCFG_ITLINE0</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																	0
0x84	<b>SYSCFG_ITLINE1</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.	PVMOUT	
	Reset value																																0	0
0x88	<b>SYSCFG_ITLINE2</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.	RTC	
	Reset value																																0	0

Table 47. 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		
0x8C	SYSCFG_ITLINE3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																		FLASH_ECC
0x90	SYSCFG_ITLINE4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRS	
	Reset value																																		RCC
0x94	SYSCFG_ITLINE5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI1	
	Reset value																																		EXTI0
0x98	SYSCFG_ITLINE6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI3	
	Reset value																																		EXTI2
0x9C	SYSCFG_ITLINE7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTI5	
	Reset value																							EXTI15	EXTI14	EXTI13	EXTI12	EXTI11	EXTI10	EXTI9	EXTI8	EXTI7	EXTI6	EXTI5	
0xA0	SYSCFG_ITLINE8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB	
	Reset value																																		UCPD2
0xA4	SYSCFG_ITLINE9	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1	
	Reset value																																		DMA1_CH1
0xA8	SYSCFG_ITLINE10	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMA1_CH3	
	Reset value																																		DMA1_CH2
0xAC	SYSCFG_ITLINE11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMA2_CH5	
	Reset value																								DMA2_CH4	DMA2_CH3	DMA2_CH2	DMA2_CH1	DMA1_CH7	DMA1_CH6	DMA1_CH5	DMA1_CH4	DMA1_CH3	DMA1_CH2	DMA1_CH1
0xB0	SYSCFG_ITLINE12	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP3
	Reset value																																		COMP2
0xB4	SYSCFG_ITLINE13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM1_BRK	
	Reset value																																		TIM1_UPD
0xB8	SYSCFG_ITLINE14	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM1_TRG
	Reset value																																		TIM1_CC
0xBC	SYSCFG_ITLINE15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM2
	Reset value																																		
0xC0	SYSCFG_ITLINE16	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM4
	Reset value																																		



Table 47. SYSCFG register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xC4	SYSCFG_ITLINE17	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xC8	SYSCFG_ITLINE18	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xCC	SYSCFG_ITLINE19	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xD0	SYSCFG_ITLINE20	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xD4	SYSCFG_ITLINE21	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xD8	SYSCFG_ITLINE22	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xDC	SYSCFG_ITLINE23	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xE0	SYSCFG_ITLINE24	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xE4	SYSCFG_ITLINE25	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xE8	SYSCFG_ITLINE26	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xEC	SYSCFG_ITLINE27	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xF0	SYSCFG_ITLINE28	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xF4	SYSCFG_ITLINE29	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xF8	SYSCFG_ITLINE30	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0xFC	SYSCFG_ITLINE31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 60](#) for the register boundary addresses.



## 9 Interconnect matrix

### 9.1 Introduction

Several peripherals have direct connections between them.

This allows autonomous communication and/or synchronization between peripherals, saving CPU resources thus power consumption.

In addition, these hardware connections remove software latency and allow design of predictable systems.

Depending on peripherals, these interconnections can operate in Run, Sleep, Low-power run, Low-power sleep, Stop 0, and Stop 1 modes.

For availability of peripherals on different STM32G0x1 products, refer to [Section 1.4: Availability of peripherals](#).

### 9.2 Connection summary

Table 48. Interconnect matrix<sup>(1)(2)</sup>

Source	Destination																
	TIM1	TIM2	TIM3	TIM4	TIM14	TIM15	TIM16	TIM17	LPTIM1	LPTIM2	ADC	DAC	DMAMUX	COMP1	COMP2	COMP3	IRTIM
TIM1	-	9.3.1	9.3.1	9.3.1	-	-	-	-	-	-	9.3.2	9.3.4	-	9.3.7	9.3.7	9.3.7	-
TIM2	9.3.1	-	9.3.1	9.3.1	-	9.3.1	-	-	-	-	9.3.2	9.3.4	-	9.3.7	9.3.7	9.3.7	-
TIM3	9.3.1	9.3.1	-	-	-	9.3.1	-	-	-	-	9.3.2	9.3.4	-	9.3.7	9.3.7	9.3.7	-
TIM4	9.3.1	9.3.1	-	-	-	9.3.1	-	-	-	-	9.3.2	9.3.4	-	9.3.7	9.3.7	9.3.7	-
TIM14	-	9.3.1	9.3.1	9.3.1	-	-	-	-	-	-	-	-	9.3.12	-	-	-	-
TIM15	9.3.1	9.3.1	9.3.1	9.3.1	-	-	-	-	-	-	9.3.2	9.3.4	-	-	-	-	-
TIM16	-	-	-	-	-	9.3.1	-	-	-	-	-	-	-	-	-	-	9.3.11
TIM17	9.3.1	-	-	-	-	9.3.1	-	-	-	-	-	-	-	-	-	-	9.3.11
TIM6	-	-	-	-	-	-	-	-	-	-	9.3.2	9.3.4	-	-	-	-	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	9.3.4	-	-	-	-	-
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	9.3.4	9.3.12	-	-	-	-
LPTIM2	-	-	-	-	-	-	-	-	-	-	-	9.3.4	9.3.12	-	-	-	-
USART1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9.3.11
USART4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9.3.11
ADC	9.3.3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
T. sensor	-	-	-	-	-	-	-	-	-	-	9.3.8	-	-	-	-	-	-
VBAT	-	-	-	-	-	-	-	-	-	-	9.3.8	-	-	-	-	-	-



Table 48. Interconnect matrix<sup>(1)(2)</sup> (continued)

Source	Destination																
	TIM1	TIM2	TIM3	TIM4	TIM14	TIM15	TIM16	TIM17	LPTIM1	LPTIM2	ADC	DAC	DMAMUX	COMP1	COMP2	COMP3	IRTIM
VREFINT	-	-	-	-	-	-	-	-	-	-	9.3.8	-	-	-	-	-	-
HSE	-	-	-	-	9.3.5	-	-	9.3.5	-	-	-	-	-	-	-	-	-
LSE	-	9.3.5	-	-	-	-	9.3.5	-	-	-	-	-	-	-	-	-	-
LSI	-	-	-	-	-	-	9.3.5	-	-	-	-	-	-	-	-	-	-
MCO	-	-	-	-	9.3.5	-	-	9.3.5	-	-	-	-	-	-	-	-	-
MCO2	-	-	-	-	9.3.5	-	-	9.3.5	-	-	-	-	-	-	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	9.3.2	9.3.4	-	-	-	-	-
RTC and TAMP	-	-	-	-	9.3.5	-	9.3.5	-	9.3.6	9.3.6	-	-	-	-	-	-	-
COMP1	9.3.9	9.3.9	9.3.9	9.3.9	-	9.3.9	9.3.9	9.3.9	9.3.6	9.3.6	-	-	-	-	-	-	-
COMP2	9.3.9	9.3.9	9.3.9	9.3.9	-	9.3.9	9.3.9	9.3.9	9.3.6	9.3.6	-	-	-	-	-	-	-
COMP3	9.3.9	9.3.9	9.3.9	9.3.9	-	9.3.9	9.3.9	9.3.9	9.3.6	9.3.6	-	-	-	-	-	-	-
SYST ERR	9.3.10	9.3.10	9.3.10	9.3.10	-	9.3.10	9.3.10	9.3.10	-	-	-	-	-	-	-	-	-

1. Numbers in the table are links to corresponding sub-sections in [Section 9.3: Interconnection details](#).
2. The “-” symbol in grayed cells means “no interconnection”.

### 9.3 Interconnection details

#### 9.3.1 From TIM1, TIM2, TIM3, TIM4, TIM15, TIM16, and TIM17, to TIM1, TIM2, TIM3, TIM4, and TIM15

##### Purpose

Some of the TIMx timers are linked together internally for timer synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop or clock the counter of another timer configured in slave mode.

A description of the feature is provided in: [Section 22.3.19: Timer synchronization](#).

The modes of synchronization are detailed in:

- [Section 21.3.26: Timer synchronization](#) for advanced-control timer TIM1
- [Section 22.3.18: Timers and external trigger synchronization](#) for general-purpose timers TIM2/TIM3/TIM4
- [Section 25.4.20: External trigger synchronization \(TIM15 only\)](#) for general-purpose timer TIM15

### Triggering signals

The output (from master) is on signal TIMx\_TRGO (and TIMx\_TRGOx), following a configurable timer event.

With TIM14, TIM16, and TIM17 timers that do not have a trigger output, the output compare 1 is used instead.

The input (to slave) is on signals TIMx\_ITR0/ITR1/ITR2/ITR3.

The input and output signals for TIM1 are shown in [Figure 101: Advanced-control timer block diagram](#).

The possible master/slave connections are given in [Table 119: TIM1 internal trigger connection](#) and [Table 130: TIMx Internal trigger connection](#).

### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

## 9.3.2 From TIM1, TIM2, TIM3, TIM4, TIM6, TIM15, and EXTI, to ADC

### Purpose

The general-purpose timers TIM2, TIM3, TIM4, and TIM15, basic timer TIM6, advanced-control timer TIM1, and EXTI can be used to generate an ADC triggering event.

TIMx synchronization is described in: [Section 21.3.27: ADC synchronization](#).

ADC synchronization is described in: [Section 15.4: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN\)](#).

### Triggering signals

The output (from timer) is on signal TIMx\_TRGO, TIMx\_TRGO2 or TIMx\_CCx event.

The input (to ADC) is on the TRG[7:0] signal.

The connection between timers and ADC is provided in [Table 73: External triggers](#).

### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

## 9.3.3 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 15.7: Analog window watchdogs](#).

Trigger settings on the timer are provided in: [Section 21.3.4: External trigger input](#).

### Triggering signals

The output (from ADC) is on signals ADC\_AWDx\_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, Sleep, Low-power run, and Low-power sleep power modes.

## 9.3.4 From TIM1, TIM2, TIM3, TIM4, TIM6, TIM7, TIM15, LPTIM1, LPTIM2, and EXTI, to DAC

### Purpose

General-purpose timers TIM2/TIM3/TIM4/TIM15, basic timers TIM6, TIM7, low-power timers LPTIM1/LPTIM2, and advanced control timer TIM1 can trigger a DAC conversion.

### Triggering signals

The TIMx\_TRGO output of each timer is directly connected to corresponding DAC input.

Selection of DAC triggering input is provided in [Section 16.4.7: DAC trigger selection](#) (single and dual mode).

### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

## 9.3.5 From HSE, LSE, LSI, MCO, MCO2, RTC and TAMP, to TIM2, TIM14, TIM16, and TIM17

### Purpose

External clocks (HSE, LSE), internal clock (LSI), microcontroller output clock (MCO and MCO2), RTC clock, RTC wake-up interrupt, and GPIO can be selected as inputs to capture channel 1 of some of TIM14/16/TIM17 timers.

The timers allow calibrating or precisely measuring internal clocks such as HSI16 or LSI, using accurate clocks such as LSE or HSE/32 for timing reference. See details in [Section 5.2.16: Internal/external clock measurement with TIM14/TIM16/TIM17](#).

When low-speed external (LSE) oscillator is used, no additional hardware connections are required.

External clock LSE can be used as input to general-purpose timers (TIM2) on TIM2\_ETR input, see [Section 22.4.25: TIM2 alternate function option register 1 \(TIM2\\_AF1\)](#).

### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

### 9.3.6 From RTC, TAMP, COMP1, COMP2, and COMP3 to LPTIM1 and LPTIM2

#### Purpose

RTC alarm A/B, TAMP1/2 input detection, COMP1/2\_OUT and GPIO alternate function can be used as trigger to start LPTIM counters LPTIM1/2.

#### Triggering signals

This trigger feature is described in [Section 26.4.7: Trigger multiplexer](#) (and following sections).

The input selection is described in [Table 138: LPTIM1 external trigger connection](#) and [Table 139: LPTIM2 external trigger connection](#).

#### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, Low-power sleep, Stop 0, and Stop 1 power modes.

### 9.3.7 From TIM1, TIM2, TIM3, TIM4, and TIM15, to COMP1, COMP2, and COMP3

#### Purpose

Advanced-control timer TIM1 and general-purpose timers TIM2, TIM3, TIM4, and TIM15 can be used as blanking window input to COMP1, COMP2, and COMP3.

The blanking function is described in [Section 18.3.7: Comparator output blanking function](#).

The blanking sources are given in:

- [Section 18.6.1: Comparator 1 control and status register \(COMP1\\_CSR\)](#) bits 20:18 BLANKING[2:0]
- [Section 18.6.2: Comparator 2 control and status register \(COMP2\\_CSR\)](#) bits 20:18 BLANKING[2:0]

#### Triggering signals

Timer output signal TIMx\_OCx are the inputs to blanking source of COMP1/COMP2/COMP3.

#### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

### 9.3.8 From internal analog sources to ADC

#### Purpose

Internal temperature sensor output voltage  $V_{TS}$ , internal reference voltage  $V_{REFINT}$  and  $V_{BAT}$  monitoring channel are connected to ADC input channels.

More information is in:

- [Section 15.2: ADC main features](#)
- [Section 15.3.8: Channel selection \(CHSEL, SCANDIR, CHSELRMOD\)](#)
- [Figure 15.9: Temperature sensor and internal reference voltage](#)
- [Figure 15.10: Battery voltage monitoring](#)

#### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

### 9.3.9 From COMP1, COMP2, and COMP3 to TIM1, TIM2, TIM3, TIM4, TIM15, TIM16, and TIM17

#### Purpose

COMP1, COMP2, and COMP3 comparator outputs can be connected to input capture or TIMx\_ETR inputs of TIM1, TIM2, TIM3, or TIM4.

The connection to ETR is described in [Section 21.3.4: External trigger input](#).

COMP1, COMP2, and COMP3 comparator outputs can also act as TIMx\_BKIN or TIMx\_BKIN2 break input signals for TIM1, TIM15, TIM16, and TIM17, through selecting GPIO alternate function using open drain connection of I/O. See [Section 21.3.17: Bidirectional break inputs](#).

The possible connections are given in:

- [Section 21.4.23: TIM1 option register 1 \(TIM1\\_OR1\)](#)
- [Section 21.4.28: TIM1 Alternate function register 2 \(TIM1\\_AF2\)](#)
- [Section 22.4.22: TIM2 option register 1 \(TIM2\\_OR1\)](#)
- [Section 25.3: TIM16/TIM17 main features](#)

#### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

### 9.3.10 From system errors to TIM1, TIM2, TIM3, TIM4, TIM15, TIM16, and TIM17

#### Purpose

CSS, CPU hardfault, RAM parity error, FLASH ECC double error detection, PVD can generate system errors in the form of timer break toward TIM1, TIM2, TIM3, TIM4, TIM15, TIM16, and TIM17.

The purpose of the break function is to protect power switches driven by PWM signals from the timers.

List of possible source of break are described in:

- [Section 21.3.16: Using the break function](#) (TIM1)
- [Section 25.4.13: Using the break function](#) (TIM15/TIM16/TIM17)
- [Figure 236: TIM15 block diagram](#)
- [Figure 237: TIM16/TIM17 block diagram](#)

#### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

### 9.3.11 From TIM16, TIM17, USART1, and USART4, to IRTIM

#### Purpose

TIMx\_OC1 output channel of TIM16 or TIM17 timers, associated with USART1 or USART4 transmission signal, can generate the infrared output waveform.

The functionality is described in [Section 27: Infrared interface \(IRTIM\)](#).

#### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

### 9.3.12 From TIM14, LPTIM1, and LPTIM2, to DMAMUX

#### Purpose

TIM14 general-purpose timer, LPTIM1 and LPTIM2 low-power timers, and EXTI, can be used as triggering event to DMAMUX.

#### Relevant power modes

These interconnections operate in Run, Sleep, Low-power run, and Low-power sleep power modes.

## 10 Direct memory access controller (DMA)

### 10.1 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.

Refer to [Section 10.3](#) for information on DMA implementation.

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.

### 10.2 DMA main features

- Single AHB master
- Peripheral-to-memory, memory-to-peripheral, memory-to-memory and peripheral-to-peripheral data transfers
- Access, as source and destination, to on-chip memory-mapped devices such as flash memory, SRAM, and AHB and APB peripherals
- All DMA channels independently configurable:
  - Each channel is associated either with a DMA request signal coming from a peripheral, or with a software trigger in memory-to-memory transfers. This configuration is done by software.
  - Priority between the requests is programmable by software (four levels per channel: very high, high, medium, low) and by hardware in case of equality (such as request to channel 1 has priority over request to channel 2).
  - Transfer size of source and destination are independent (byte, half-word, word), emulating packing and unpacking. Source and destination addresses must be aligned on the data size.
  - Support of transfers from/to peripherals to/from memory with circular buffer management
  - Programmable number of data to be transferred: 0 to  $2^{16} - 1$
- Generation of an interrupt request per channel. Each interrupt request is caused from any of the three DMA events: transfer complete, half transfer, or transfer error.

## 10.3 DMA implementation

### 10.3.1 DMA

The devices incorporate one or two DMA controller instances. The following implementation table shows the number of DMA channels for either instance. A dash indicates that the instance is not implemented.

**Table 49. DMA implementation**

Number of channels	STM32G031xx STM32G041xx	STM32G051xx STM32G061xx STM32G071xx STM32G081xx	STM32G0B1xx STM32G0C1xx
DMA1	5	7	7
DMA2	-	-	5

### 10.3.2 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.

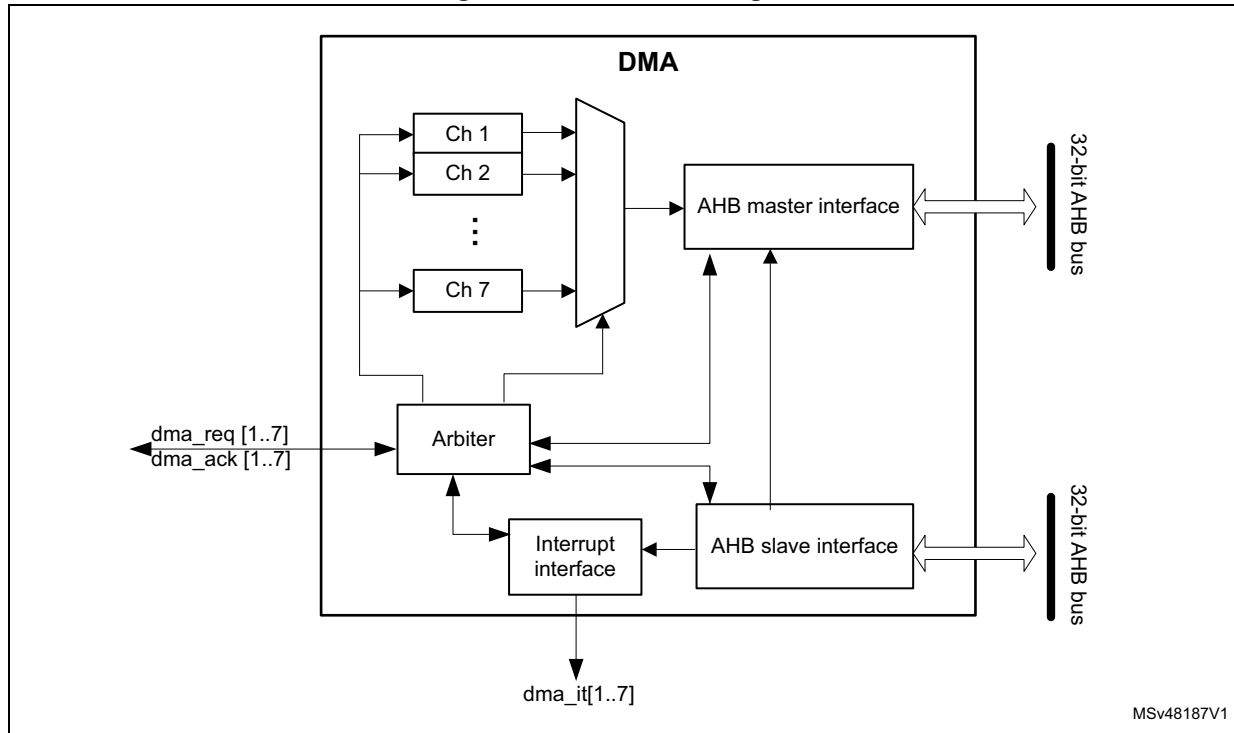


## 10.4 DMA functional description

### 10.4.1 DMA block diagram

The DMA block diagram is shown in the figure below.

Figure 22. 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.

### 10.4.2 DMA pins and internal signals

Table 50. 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

### 10.4.3 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\_CPAR $x$  or DMA\_CMAR $x$  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\_CPAR $x$  or DMA\_CMAR $x$  register.
- post-decrementing of the programmed DMA\_CNDTR $x$  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\_CNDTR $x$  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.

#### 10.4.4 DMA arbitration

The DMA arbiter manages the priority between the different channels.

When an active channel *x* is granted by the arbiter (hardware requested or software triggered), a single DMA transfer is issued (such as a AHB 'read followed by write' transfer of a single data). Then, the arbiter considers again the set of active channels and selects the one with the highest priority.

The priorities are managed in two stages:

- software: priority of each channel is configured in the DMA\_CCRx register, to one of the four different levels:
  - very high
  - high
  - medium
  - low
- hardware: if two requests have the same software priority level, the channel with the lowest index gets priority. For example, channel 2 gets priority over channel 4.

When a channel *x* is programmed for a block transfer in memory-to-memory mode, re arbitration is considered between each single DMA transfer of this channel *x*. Whenever there is another concurrent active requested channel, the DMA arbiter automatically alternates and grants the other highest-priority requested channel, which may be of lower priority than the memory-to-memory channel.

#### 10.4.5 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 the active state is an enabled channel (read DMA\_CCRx.EN = 1). An active channel x is a channel that must have been enabled by the software (DMA\_CCRx.EN set to 1) and afterwards with no occurred transfer error (DMA\_ISR.TEIFx = 0). In case there is a transfer error, the channel is automatically disabled by hardware (DMA\_CCRx.EN = 0).

The three following use cases may happen:

- Suspend and resume a channel

This corresponds to the two following actions:

- An active channel is disabled by software (writing DMA\_CCRx.EN = 0 whereas DMA\_CCRx.EN = 1).
- The software enables the channel again (DMA\_CCRx.EN set to 1) without reconfiguring the other channel registers (such as DMA\_CNDTRx, DMA\_CPARx and DMA\_CMARx).

This case is not supported by the DMA hardware, that does not guarantee that the remaining data transfers are performed correctly.

- Stop and abort a channel

If the application does not need any more the channel, this active channel can be disabled by software. The channel is stopped and aborted but the DMA\_CNDTRx register content may not correctly reflect the remaining data transfers versus the aborted source and destination buffer/register.

- Abort and restart a channel

This corresponds to the software sequence: disable an active channel, then reconfigure the channel and enable it again.

This is supported by the hardware if the following conditions are met:

- The application guarantees that, when the software is disabling the channel, a DMA data transfer is not occurring at the same time over its master port. For example, the application can first disable the peripheral in DMA mode, in order to ensure that there is no pending hardware DMA request from this peripheral.
- The software must operate separated write accesses to the same DMA\_CCRx register: First disable the channel. Second reconfigure the channel for a next block transfer including the DMA\_CCRx if a configuration change is needed. There are read-only DMA\_CCRx register fields when DMA\_CCRx.EN=1. Finally enable again the channel.

When a channel transfer error occurs, the EN bit of the DMA\_CCRx register is cleared by hardware. This EN bit cannot be set again by software to reactivate the channel x, until the TEIFx bit of the DMA\_ISR register is set.

### **Circular mode (in memory-to-peripheral/peripheral-to-memory transfers)**

The circular mode is available to handle circular buffers and continuous data flows (such as ADC scan mode). This feature is enabled using the CIRC bit in the DMA\_CCRx register.

*Note:*

*The circular mode must not be used in memory-to-memory mode. Before enabling a channel in circular mode (CIRC = 1), the software must clear the MEM2MEM bit of the DMA\_CCRx register. When the circular mode is activated, the amount of data to transfer is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.*

*In order to stop a circular transfer, the software needs to stop the peripheral from generating DMA requests (such as quit the ADC scan mode), before disabling the DMA channel. The software must explicitly program the DMA\_CNDTRx value before starting/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 the 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 the 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 the 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.

### 10.4.6 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 51. 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 the AHB master bus).
- To write the byte 0xAB, the DMA controller sets the HWDATA bus to 0xABABABAB with a byte data size (HSIZE = Byte in the AHB master bus).

Assuming the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take into account the HSIZE data, any AHB byte or half-word transfer is changed into a 32-bit APB transfer as described below:

- An AHB byte write transfer of 0xB0 to one of the 0x0, 0x1, 0x2, or 0x3 addresses, is converted to an APB word write transfer of 0xB0B0B0B0 to the 0x0 address.
- An AHB half-word write transfer of 0xB1B0 to the 0x0 or 0x2 addresses is converted to an APB word write transfer of 0xB1B0B1B0 to the 0x0 address.

#### 10.4.7 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 cannot be set again by software (channel x reactivated) until the TEIFx bit of the DMA\_ISR register is cleared (by setting the CTEIFx bit of the DMA\_IFCR register).

When the software is notified with a transfer error over a channel, which involves a peripheral, the software has first to stop this peripheral in DMA mode, in order to disable any pending or future DMA request. Then software may normally reconfigure both DMA and the peripheral in DMA mode for a new transfer.



## 10.5 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 52. 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	-

## 10.6 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).

### 10.6.1 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.	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TEIF7**: Transfer error (TE) flag for channel 7

- 0: No TE event
- 1: A TE event occurred.

Bit 26 **HTIF7**: Half transfer (HT) flag for channel 7

- 0: No HT event
- 1: An HT event occurred.

Bit 25 **TCIF7**: Transfer complete (TC) flag for channel 7

- 0: No TC event
- 1: A TC event occurred.

Bit 24 **GIF7**: Global interrupt flag for channel 7

- 0: No TE, HT, or TC event
- 1: A TE, HT, or TC event occurred.

- Bit 23 **TEIF6**: Transfer error (TE) flag for channel 6  
0: No TE event  
1: A TE event occurred.
- Bit 22 **HTIF6**: Half transfer (HT) flag for channel 6  
0: No HT event  
1: An HT event occurred.
- Bit 21 **TCIF6**: Transfer complete (TC) flag for channel 6  
0: No TC event  
1: A TC event occurred.
- Bit 20 **GIF6**: Global interrupt flag for channel 6  
0: No TE, HT, or TC event  
1: A TE, HT, or TC event occurred.
- Bit 19 **TEIF5**: Transfer error (TE) flag for channel 5  
0: No TE event  
1: A TE event occurred.
- Bit 18 **HTIF5**: Half transfer (HT) flag for channel 5  
0: No HT event  
1: An HT event occurred.
- Bit 17 **TCIF5**: Transfer complete (TC) flag for channel 5  
0: No TC event  
1: A TC event occurred.
- Bit 16 **GIF5**: global interrupt flag for channel 5  
0: No TE, HT, or TC event  
1: A TE, HT, or TC event occurred.
- Bit 15 **TEIF4**: Transfer error (TE) flag for channel 4  
0: No TE event  
1: A TE event occurred.
- Bit 14 **HTIF4**: Half transfer (HT) flag for channel 4  
0: No HT event  
1: An HT event occurred.
- Bit 13 **TCIF4**: Transfer complete (TC) flag for channel 4  
0: No TC event  
1: A TC event occurred.
- Bit 12 **GIF4**: global interrupt flag for channel 4  
0: No TE, HT, or TC event  
1: A TE, HT, or TC event occurred.
- Bit 11 **TEIF3**: Transfer error (TE) flag for channel 3  
0: No TE event  
1: A TE event occurred.
- Bit 10 **HTIF3**: Half transfer (HT) flag for channel 3  
0: No HT event  
1: An HT event occurred.
- Bit 9 **TCIF3**: Transfer complete (TC) flag for channel 3  
0: No TC event  
1: A TC event occurred.

- Bit 8 **GIF3**: Global interrupt flag for channel 3  
0: No TE, HT, or TC event  
1: A TE, HT, or TC event occurred.
- Bit 7 **TEIF2**: Transfer error (TE) flag for channel 2  
0: No TE event  
1: A TE event occurred.
- Bit 6 **HTIF2**: Half transfer (HT) flag for channel 2  
0: No HT event  
1: An HT event occurred.
- Bit 5 **TCIF2**: Transfer complete (TC) flag for channel 2  
0: No TC event  
1: A TC event occurred.
- Bit 4 **GIF2**: Global interrupt flag for channel 2  
0: No TE, HT, or TC event  
1: A TE, HT, or TC event occurred.
- Bit 3 **TEIF1**: Transfer error (TE) flag for channel 1  
0: No TE event  
1: A TE event occurred.
- Bit 2 **HTIF1**: Half transfer (HT) flag for channel 1  
0: No HT event  
1: An HT event occurred.
- Bit 1 **TCIF1**: Transfer complete (TC) flag for channel 1  
0: No TC event  
1: A TC event occurred.
- Bit 0 **GIF1**: Global interrupt flag for channel 1  
0: No TE, HT, or TC event  
1: A TE, HT, or TC event occurred.

### 10.6.2 DMA interrupt flag clear register (DMA\_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

Setting the global clear bit CGIF<sub>x</sub> of the channel x in this DMA\_IFCR register, causes the DMA hardware to clear the corresponding GIF<sub>x</sub> bit and any individual flag among TEIF<sub>x</sub>, HTIF<sub>x</sub>, TCIF<sub>x</sub>, in the DMA\_ISR register.

Setting any individual clear bit among CTEIF<sub>x</sub>, CHTIF<sub>x</sub>, CTCIF<sub>x</sub> in this DMA\_IFCR register, causes the DMA hardware to clear the corresponding individual flag and the global flag GIF<sub>x</sub> 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.	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:28 Reserved, must be kept at reset value.

- Bit 27 **CTEIF7**: Transfer error flag clear for channel 7
- Bit 26 **CHTIF7**: Half transfer flag clear for channel 7
- Bit 25 **CTCIF7**: Transfer complete flag clear for channel 7
- Bit 24 **CGIF7**: Global interrupt flag clear for channel 7
- Bit 23 **CTEIF6**: Transfer error flag clear for channel 6
- Bit 22 **CHTIF6**: Half transfer flag clear for channel 6
- Bit 21 **CTCIF6**: Transfer complete flag clear for channel 6
- Bit 20 **CGIF6**: Global interrupt flag clear for channel 6
- Bit 19 **CTEIF5**: Transfer error flag clear for channel 5
- Bit 18 **CHTIF5**: Half transfer flag clear for channel 5
- Bit 17 **CTCIF5**: Transfer complete flag clear for channel 5
- Bit 16 **CGIF5**: Global interrupt flag clear for channel 5
- Bit 15 **CTEIF4**: Transfer error flag clear for channel 4
- Bit 14 **CHTIF4**: Half transfer flag clear for channel 4
- Bit 13 **CTCIF4**: Transfer complete flag clear for channel 4
- Bit 12 **CGIF4**: Global interrupt flag clear for channel 4
- Bit 11 **CTEIF3**: Transfer error flag clear for channel 3
- Bit 10 **CHTIF3**: Half transfer flag clear for channel 3
- Bit 9 **CTCIF3**: Transfer complete flag clear for channel 3
- Bit 8 **CGIF3**: Global interrupt flag clear for channel 3
- Bit 7 **CTEIF2**: Transfer error flag clear for channel 2
- Bit 6 **CHTIF2**: Half transfer flag clear for channel 2
- Bit 5 **CTCIF2**: Transfer complete flag clear for channel 2
- Bit 4 **CGIF2**: Global interrupt flag clear for channel 2
- Bit 3 **CTEIF1**: Transfer error flag clear for channel 1
- Bit 2 **CHTIF1**: Half transfer flag clear for channel 1
- Bit 1 **CTCIF1**: Transfer complete flag clear for channel 1
- Bit 0 **CGIF1**: Global interrupt flag clear for channel 1

### 10.6.3 DMA channel x configuration register (DMA\_CCRx)

Address offset: 0x08 + 0x14 \* (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

The register fields/bits MEM2MEM, PL[1:0], MSIZE[1:0], PSIZE[1:0], MINC, PINC, and DIR are read-only when EN = 1.

The states of MEM2MEM and CIRC bits must not be both high at the same time.

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.*

### 10.6.4 DMA channel x number of data to transfer register (DMA\_CNDTRx)

Address offset: 0x0C + 0x14 \* (x - 1), (x = 1 to 7)

Reset value: 0x0000 0000

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).*

### 10.6.5 DMA channel x peripheral address register (DMA\_CPARx)

Address offset:  $0x10 + 0x14 * (x - 1)$ , (x = 1 to 7)

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).*



### 10.6.6 DMA channel x memory address register (DMA\_CMARx)

Address offset:  $0x14 + 0x14 * (x - 1)$ , (x = 1 to 7)

Reset value: 0x0000 0000

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).*

### 10.6.7 DMA register map

Table 53. 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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x000	DMA_ISR	Res	Res	Res	Res	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1			
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x004	DMA_IFCR	Res	Res	Res	Res	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1			
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x008	DMA_CCR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN						
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x00C	DMA_CNDTR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDTR[15:0]																	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x010	DMA_CPAR1	PA[31:0]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x014	DMA_CMAR1	MA[31:0]																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x018	Reserved	Reserved.																																		

Table 53. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	DMA_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x020	DMA_CNDTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x028	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x02C	Reserved	Reserved.																															
0x030	DMA_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x034	DMA_CNDTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x040	Reserved	Reserved.																															
0x044	DMA_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x048	DMA_CNDTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x050	DMA_CMAR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x054	Reserved	Reserved.																															
0x058	DMA_CCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x05C	DMA_CNDTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x064	DMA_CMAR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x068	Reserved	Reserved.																															
0x06C	DMA_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 53. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x070	DMA_CNDTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_CPAR6	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_CMAR6	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved	Reserved																															
0x080	DMA_CCR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEM2MEM	PL[1:0]	MSIZE[1:0]	PSIZE[1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x084	DMA_CNDTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDTR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x088	DMA_CPAR7	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08C	DMA_CMAR7	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

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

# 11 DMA request multiplexer (DMAMUX)

## 11.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controllers of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in [Section 11.3.1](#).

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 11.3.2](#).

## 11.2 DMAMUX main features

- Up to 12-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 73 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

## 11.3 DMAMUX implementation

### 11.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

**Table 54. DMAMUX instantiation**

Feature	DMAMUX
Number of DMAMUX output request channels	12/7/5 <sup>(1)</sup>
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	Up to 73

1. 12 for STM32G0B1xx and STM32G0C1xx, seven for STM32G071xx and STM32G081xx as well as for STM32G051xx and STM32G061xx, and five for STM32G031xx and STM32G041xx devices.

### 11.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

DMAMUX1 is used with DMA1 and DMA2 (on devices supporting DMA2) as follows:

- The DMAMUX1 channels 0 to 6 are connected to DMA1 channels 1 to 7.
- The DMAMUX1 channels 7 to 11 are connected to DMA2 channels 1 to 5.

**Table 55. DMAMUX: assignment of multiplexer inputs to resources**

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux_req_gen0	27	TIM2_CH2	53	USART2_TX
2	dmamux_req_gen1	28	TIM2_CH3	54	USART3_RX
3	dmamux_req_gen2	29	TIM2_CH4	55	USART3_TX
4	dmamux_req_gen3	30	TIM2_TRIG	56	USART4_RX
5	ADC	31	TIM2_UP	57	USART4_TX
6	AES_IN	32	TIM3_CH1	58	UCPD1_RX
7	AES_OUT	33	TIM3_CH2	59	UCPD1_TX
8	DAC_Channel1	34	TIM3_CH3	60	UCPD2_RX
9	DAC_Channel2	35	TIM3_CH4	61	UCPD2_TX
10	I2C1_RX	36	TIM3_TRIG	62	I2C3_RX
11	I2C1_TX	37	TIM3_UP	63	I2C3_TX
12	I2C2_RX	38	TIM6_UP	64	LPUART2_RX
13	I2C2_TX	39	TIM7_UP	65	LPUART2_TX
14	LPUART_RX	40	TIM15_CH1	66	SPI3_RX
15	LPUART_TX	41	TIM15_CH2	67	SPI3_TX
16	SPI1_RX	42	TIM15_TRIG_COM	68	TIM4_CH1
17	SPI1_TX	43	TIM15_UP	69	TIM4_CH2
18	SPI2_RX	44	TIM16_CH1	70	TIM4_CH3
19	SPI2_TX	45	TIM16_COM	71	TIM4_CH4
20	TIM1_CH1	46	TIM16_UP	72	TIM4_TRIG
21	TIM1_CH2	47	TIM17_CH1	73	TIM4_UP
22	TIM1_CH3	48	TIM17_COM	74	USART5_RX
23	TIM1_CH4	49	TIM17_UP	75	USART5_TX
24	TIM1_TRIG_COM	50	USART1_RX	76	USART6_RX
25	TIM1_UP	51	USART1_TX	77	USART6_TX
26	TIM2_CH1	52	USART2_RX	-	-

**Table 56. 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

**Table 56. DMAMUX: assignment of trigger inputs to resources (continued)**

Trigger input	Resource	Trigger input	Resource
6	EXTI LINE6	18	dmamux_evt2
7	EXTI LINE7	19	dmamux_evt3
8	EXTI LINE8	20	LPTIM1_OUT
9	EXTI LINE9	21	LPTIM2_OUT
10	EXTI LINE10	22	TIM14_OC
11	EXTI LINE11	23	Reserved

**Table 57. DMAMUX: assignment of synchronization inputs to resources**

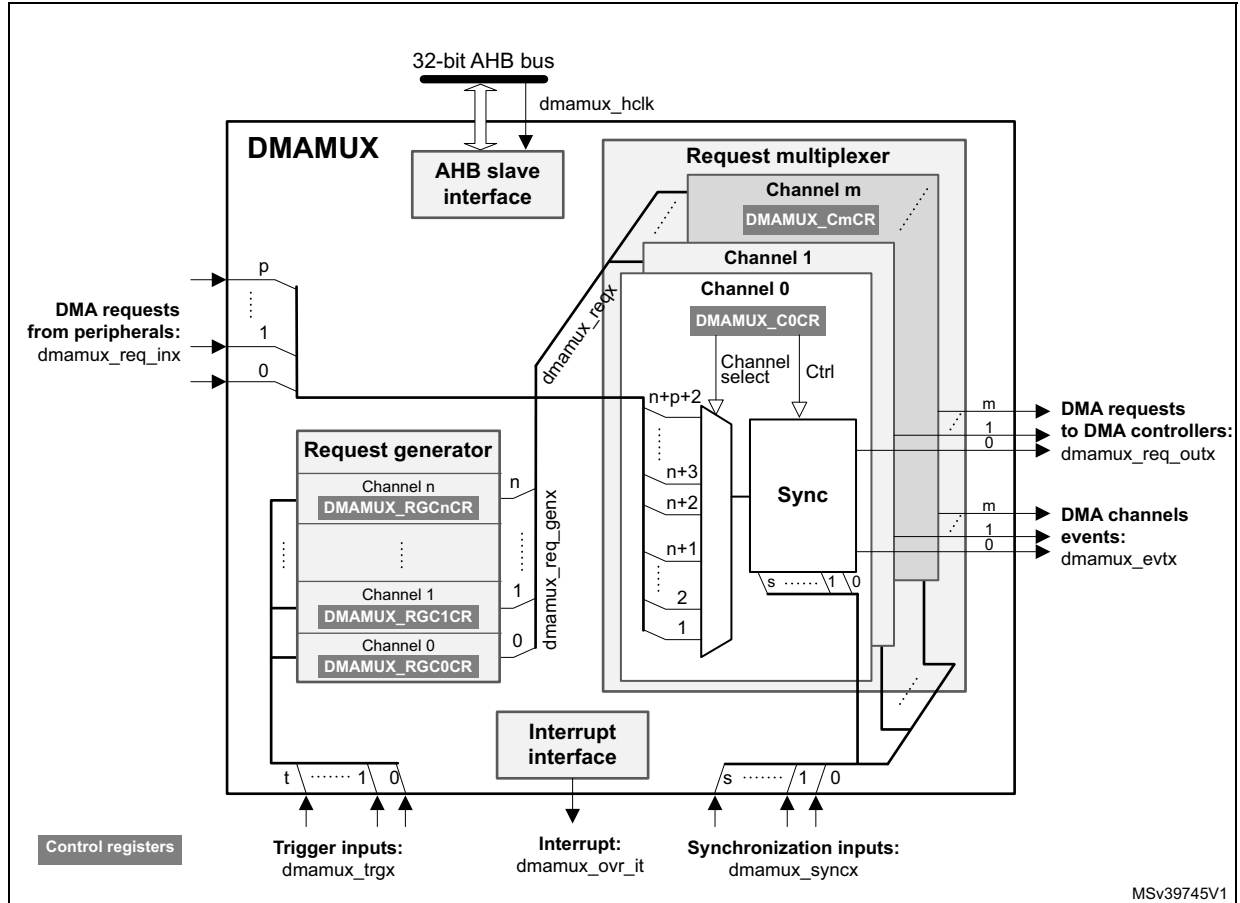
Sync. input	Resource	Sync. input	Resource
0	EXTI LINE0	12	EXTI LINE12
1	EXTI LINE1	13	EXTI LINE13
2	EXTI LINE2	14	EXTI LINE14
3	EXTI LINE3	15	EXTI LINE15
4	EXTI LINE4	16	dmamux_evt0
5	EXTI LINE5	17	dmamux_evt1
6	EXTI LINE6	18	dmamux_evt2
7	EXTI LINE7	19	dmamux_evt3
8	EXTI LINE8	20	LPTIM1_OUT
9	EXTI LINE9	21	LPTIM2_OUT
10	EXTI LINE10	22	TIM14_OC
11	EXTI LINE11	23	Reserved

## 11.4 DMAMUX functional description

### 11.4.1 DMAMUX block diagram

Figure 23 shows the DMAMUX block diagram.

Figure 23. DMAMUX block diagram



DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux\_reqx) from peripherals (dmamux\_req\_inx) and from channels of the DMAMUX request generator sub-block (dmamux\_req\_genx)
- DMAMUX request outputs to channels of DMA controllers (dmamux\_req\_outx)
- Internal or external signals to DMA request trigger inputs (dmamux\_trgx)
- Internal or external signals to synchronization inputs (dmamux\_syncx)



## 11.4.2 DMAMUX signals

Table 58 lists the DMAMUX signals.

**Table 58. 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 controllers)
dmamux_evtx	DMAMUX events outputs
dmamux_ovr_it	DMAMUX overrun interrupts

## 11.4.3 DMAMUX channels

A DMAMUX channel is a request multiplexer channel that can include, depending upon the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to a single channel of DMA controller(s).

### Channel configuration procedure

Follow the sequence below to configure a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

## 11.4.4 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 25](#), 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 24. Synchronization mode of the DMAMUX request line multiplexer channel

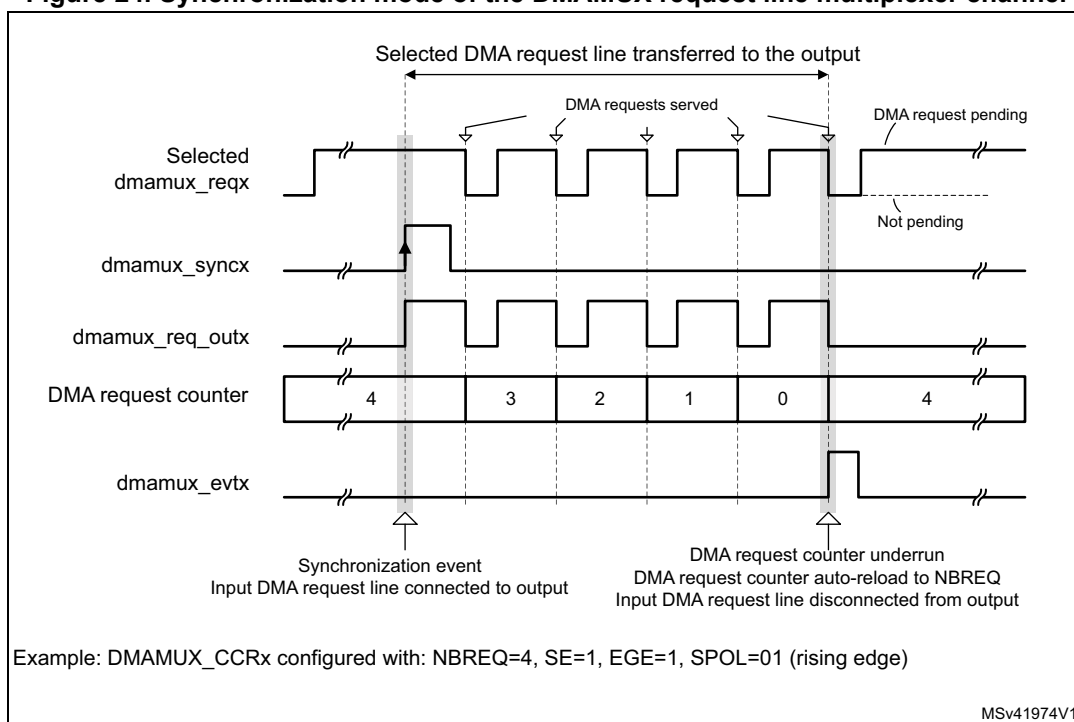
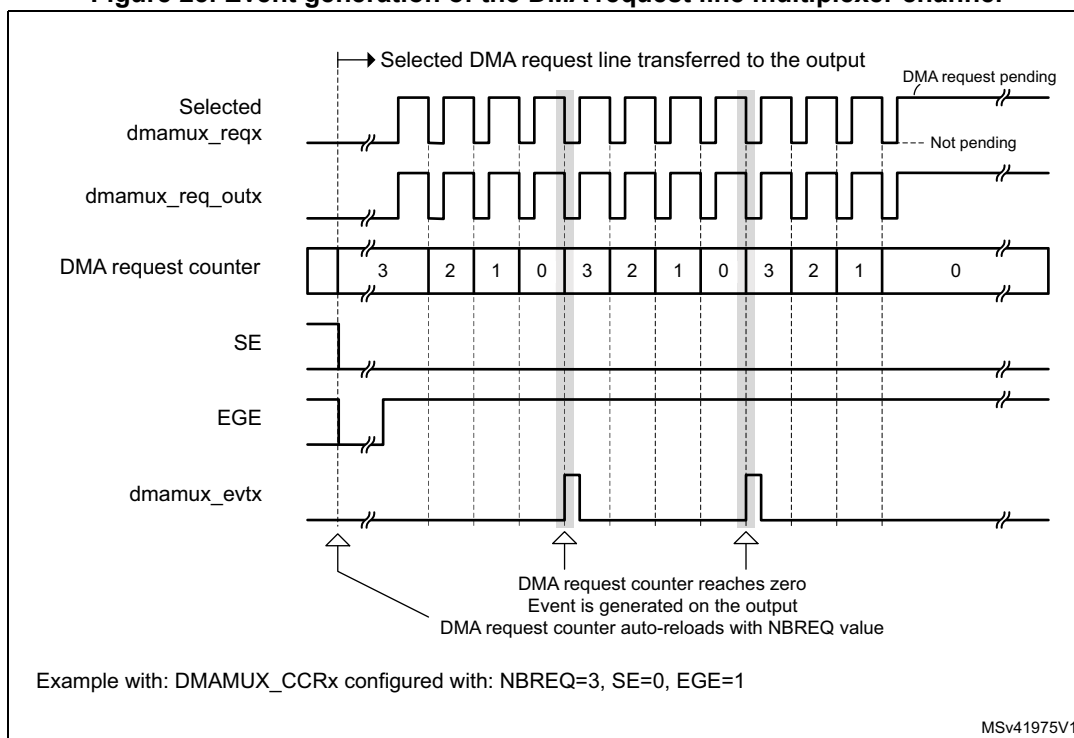


Figure 25. Event generation of the DMA request line multiplexer channel



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 24](#) and [Figure 25](#).

*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.

## 11.4.5 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.

**11.5 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 59. 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

## 11.6 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.

### 11.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX\_CxCR)

Address offset:  $0x000 + 0x04 * x$  ( $x = 0$  to  $11$ )

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]		SE		
			r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
						r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w

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 57: 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

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DMAREQ\_ID[6:0]**: DMA request identification

Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

### 11.6.2 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.	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.	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 11:0 **SOF[11:0]**: 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.

### 11.6.3 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.	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.	CSOF 11	CSOF 10	CSOF 9	CSOF 8	CSOF 7	CSOF 6	CSOF 5	CSOF 4	CSOF 3	CSOF 2	CSOF 1	CSOF 0
				w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 11:0 **CSOF[11:0]**: Clear synchronization overrun event flag

Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX\_CSR register.

### 11.6.4 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.	Res.	Res.	Res.	Res.	Res.	Res.	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.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	SIG_ID[4:0]				
							rw				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



### 11.6.5 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.	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.	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.

### 11.6.6 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.	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.	COF3	COF2	COF1	COF0
												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.

### 11.6.7 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 60. 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]	SE	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]						
	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
0x004	DMAMUX_C1CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x008	DMAMUX_C2CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x00C	DMAMUX_C3CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x010	DMAMUX_C4CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x014	DMAMUX_C5CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x018	DMAMUX_C6CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x01C	DMAMUX_C7CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x020	DMAMUX_C8CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x024	DMAMUX_C9CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x028	DMAMUX_C10CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x02C	DMAMUX_C11CR	Res	Res	Res	SYNC_ID[4:0]				NBREQ[4:0]				SPOL	[1:0]	SE	Res	Res	Res	Res	Res	Res	Res	Res	Res	EGE	SOIE	Res	DMAREQ_ID[6:0]					
	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
0x030-0x07C	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	Res	Res	Res
0x080	DMAMUX_CSR	Res	Res	Res	Res	Res	Res	Res	Res	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																																
0x084	DMAMUX_CFR	Res	Res	Res	Res	Res	Res	Res	Res	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																																
0x088 - 0x0FC	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	Res
0x100	DMAMUX_RG0CR	Res	Res	Res	Res	Res	Res	Res	Res	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 60. DMAMUX 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	
0x104	DMAMUX_RG1CR	Res	Res	Res	Res	Res	Res	Res	Res	GNBREQ[4:0]				GPOL	[1:0]	GE	Res	Res	Res	Res	Res	Res	Res	Res	OIE	Res	Res	Res	Res	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	[1:0]	0	0	Res	Res	Res	Res	Res	Res	Res	0	Res	Res	Res	Res	0	0	0	0
0x108	DMAMUX_RG2CR	Res	Res	Res	Res	Res	Res	Res	Res	GNBREQ[4:0]				GPOL	[1:0]	GE	Res	Res	Res	Res	Res	Res	Res	Res	OIE	Res	Res	Res	Res	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	[1:0]	0	0	Res	Res	Res	Res	Res	Res	Res	0	Res	Res	Res	Res	0	0	0	0
0x10C	DMAMUX_RG3CR	Res	Res	Res	Res	Res	Res	Res	Res	GNBREQ[4:0]				GPOL	[1:0]	GE	Res	Res	Res	Res	Res	Res	Res	Res	OIE	Res	Res	Res	Res	SIG_ID[4:0]				
	Reset value									0	0	0	0	0	0	[1:0]	0	0	Res	Res	Res	Res	Res	Res	Res	0	Res	Res	Res	Res	0	0	0	0
0x110 - 0x13C	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	Res	Res
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	Res	Res	Res
	Reset value																																	
0x144	DMAMUX_RGCFR	Res	Res	Res	Res	Res	Res	Res	Res	Res	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																																	
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	Res	Res	Res

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

## 12 Nested vectored interrupt controller (NVIC)

### 12.1 Main features

- 32 maskable interrupt channels (not including the sixteen Cortex®-M0+ system exceptions)
- 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.

### 12.2 SysTick calibration value register

The SysTick calibration value is set to 1000, which gives a reference time base of 1 ms with the SysTick clock set to 1 MHz.

### 12.3 Interrupt and exception vectors

[Table 61](#) is the vector table. Information pertaining to a peripheral only applies to devices containing that peripheral.

**Table 61. 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. The SRAM parity err., flash memory ECC double err., 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 SVC instruction	0x0000_002C

Table 61. Vector table<sup>(1)</sup> (continued)

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	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	8	settable	PVD / PVM	Power voltage detector interrupt (EXTI line 16) and VDDIO2 power voltage monitor interrupt (EXTI line 34)	0x0000_0044
2	9	settable	RTC / TAMP	RTC and TAMP interrupts (combined EXTI lines 19 and 21)	0x0000_0048
3	10	settable	FLASH	Flash memory global interrupt	0x0000_004C
4	11	settable	RCC / CRS	RCC and CRS global interrupt	0x0000_0050
5	12	settable	EXTI0_1	EXTI line 0 and 1 interrupt	0x0000_0054
6	13	settable	EXTI2_3	EXTI line 2 and 3 interrupt	0x0000_0058
7	14	settable	EXTI4_15	EXTI line 4 to 15 interrupt	0x0000_005C
8	15	settable	UCPD1 / UCPD2 / USB	UCPD and USB global interrupt (combined with EXTI lines 32, 33, and 36)	0x0000_0060
9	16	settable	DMA1_Channel1	DMA1 channel 1 interrupt	0x0000_0064
10	17	settable	DMA1_Channel2_3	DMA1 channel 2 and 3 interrupts	0x0000_0068
11	18	settable	DMA1_Channel4_5_6_7 / DMAMUX / DMA2_Channel1_2_3_4_5	DMA1 channel 4, 5, 6, 7, DMAMUX, DMA2 channel 1, 2, 3, 4, 5 interrupts	0x0000_006C
12	19	settable	ADC / COMP	ADC and COMP interrupts (ADC combined with EXTI 17 and 18)	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	22	settable	TIM2	TIM2 global interrupt	0x0000_007C
16	23	settable	TIM3 / TIM4	TIM3 global interrupt	0x0000_0080
17	24	settable	TIM6 / DAC / LPTIM1	TIM6, LPTIM1 and DAC global interrupt	0x0000_0084
18	25	settable	TIM7 / LPTIM2	TIM7 and LPTIM2 global interrupt	0x0000_0088
19	26	settable	TIM14	TIM14 global interrupt	0x0000_008C
20	27	settable	TIM15	TIM15 global interrupt	0x0000_0090
21	28	settable	TIM16 / FDCAN_IT0	TIM16 and FDCAN_IT0 global interrupt	0x0000_0094

Table 61. Vector table<sup>(1)</sup> (continued)

Position	Priority	Type of priority	Acronym	Description	Address
22	29	settable	TIM17 / FDCAN_IT1	TIM17 and FDCAN_IT1 global interrupt	0x0000_0098
23	30	settable	I2C1	I2C1 global interrupt (combined with EXTI 23)	0x0000_009C
24	31	settable	I2C2 / I2C3	I2C2 and I2C3 global interrupt	0x0000_00A0
25	32	settable	SPI1	SPI1 global interrupt	0x0000_00A4
26	33	settable	SPI2 / SPI3	SPI2 global interrupt	0x0000_00A8
27	34	settable	USART1	USART1 global interrupt (combined with EXTI 25)	0x0000_00AC
28	35	settable	USART2 / LPUART2	USART2 and LPUART2 global interrupt (combined with EXTI 26)	0x0000_00B0
29	36	settable	USART3 / USART4 / USART5 / USART6 / LPUART1	USART3/4/5/6 and LPUART1 global interrupt (combined with EXTI 28)	0x0000_00B4
30	37	settable	CEC	CEC global interrupt (combined with EXTI 27)	0x0000_00B8
31	38	settable	AES / RNG	AES and RNG global interrupts	0x0000_00BC

1. The grayed cells correspond to the Cortex<sup>®</sup>-M0+ system exceptions.

## 13 Extended interrupt and event controller (EXTI)

The Extended interrupt and event controller (EXTI) manages the CPU and system wake-up through configurable and direct event inputs (lines). It provides wake-up 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 wake-up requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can also be used in Run modes.

The EXTI also includes the EXTI I/O port mux.

### 13.1 EXTI main features

The EXTI main features are the following:

- System wake-up upon event on any input
- Wake-up flag and CPU interrupt generation for events not having a wake-up 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 wake-up, 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 wake-up and event generation
  - No SW trigger possibility
- I/O port selector

### 13.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block, and EXTI mux as shown in [Figure 26](#).

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 wake-up, interrupt and event outputs, and the masking of these.

The EXTI mux provides the I/O port selection on to the EXTI event signal.

Figure 26. EXTI block diagram

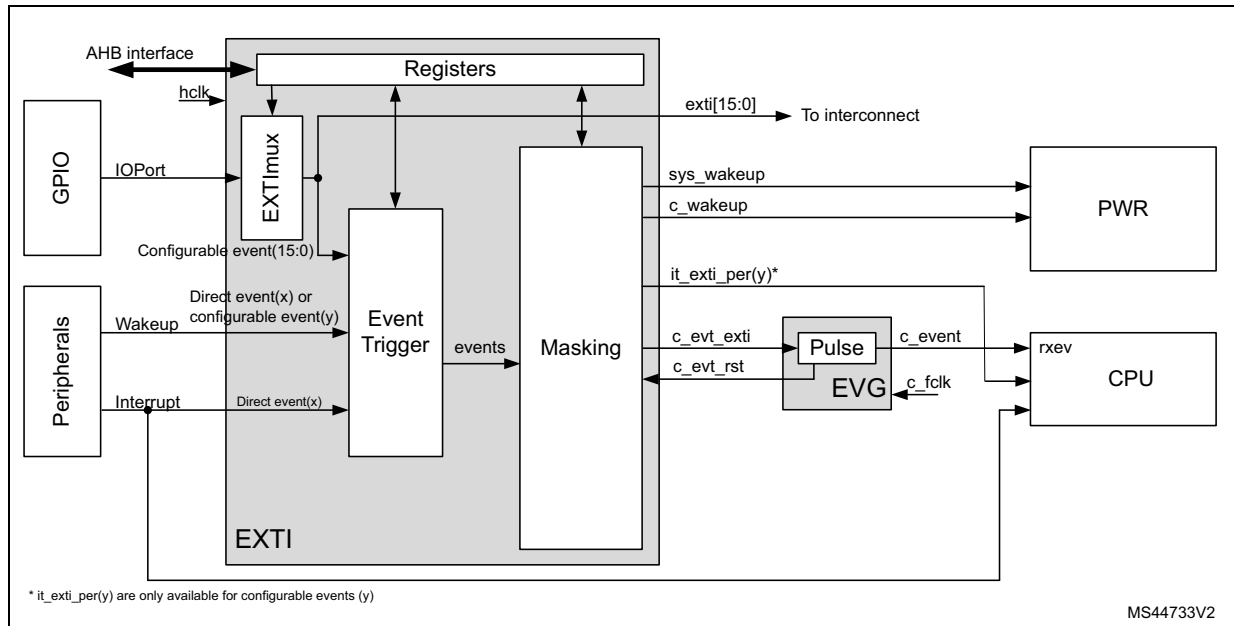


Table 62. EXTI signal overview

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 wake-up events from peripherals that do not have an associated interrupt and flag in the peripheral
Direct event(x)	I	Synchronous and asynchronous wake-up 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 wake-up request to PWR for ck_sys and hclk
c_wakeup	O	Wake-up request to PWR for CPU, synchronous to hclk

Table 63. 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



### 13.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wake-up or interrupt events when the system is in Stop mode are connected to the EXTI.

- Peripheral wake-up 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 wake-up 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 wake-up 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 wake-up signals are connected to the PWR block, and are used to wake up the system and CPU sub-system bus clocks.

## 13.3 EXTI functional description

Depending on the EXTI line type and wake-up 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 \(EXTI\\_RTISR1\)](#)
  - [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 wake-up and interrupt enable through
  - [EXTI CPU wake-up with interrupt mask register \(EXTI\\_IMR1\)](#)
  - [EXTI CPU wake-up with interrupt mask register \(EXTI\\_IMR2\)](#)
- CPU wake-up and event enable through
  - [EXTI CPU wake-up with event mask register \(EXTI\\_EMR1\)](#)
  - [EXTI CPU wake-up with event mask register \(EXTI\\_EMR2\)](#)

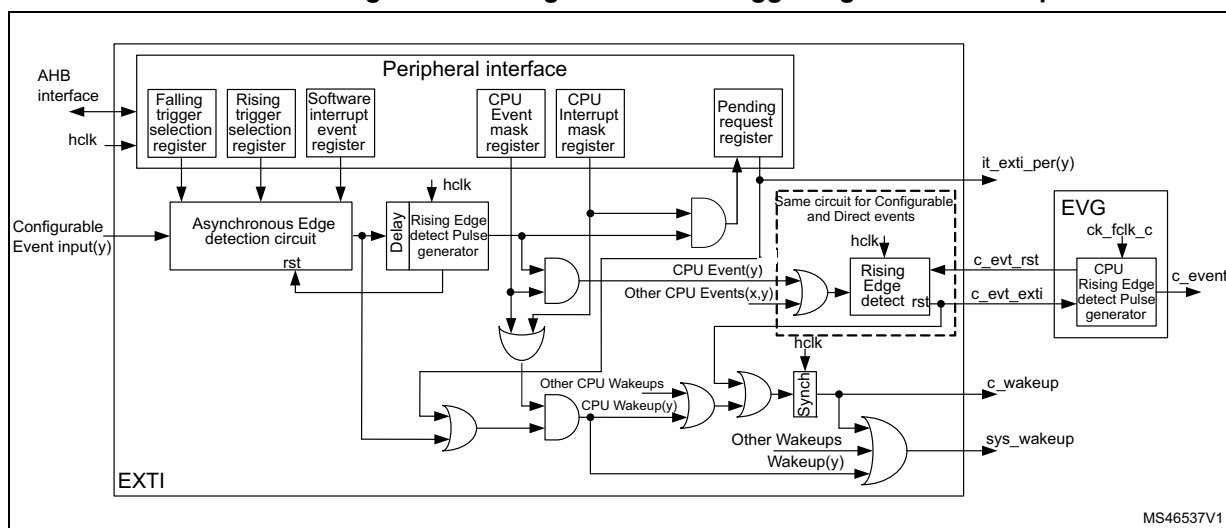
Table 64. EXTI event input configurations and register control

Event input type	Logic implementation	EXTI_RTSR1	EXTI_FTSR1	EXTI_SWIER1	EXTI_R/FPR1	EXTI_IMRx	EXTI_EMRx
Configurable	Configurable event input wake-up logic	x	x	x	x	x	x
Direct	Direct event input wake-up logic	-	-	-	-	x	x

### 13.3.1 EXTI configurable event input wake-up

Figure 27 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 wake-up event.

Figure 27. Configurable event trigger logic CPU wake-up



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 wake-up 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.

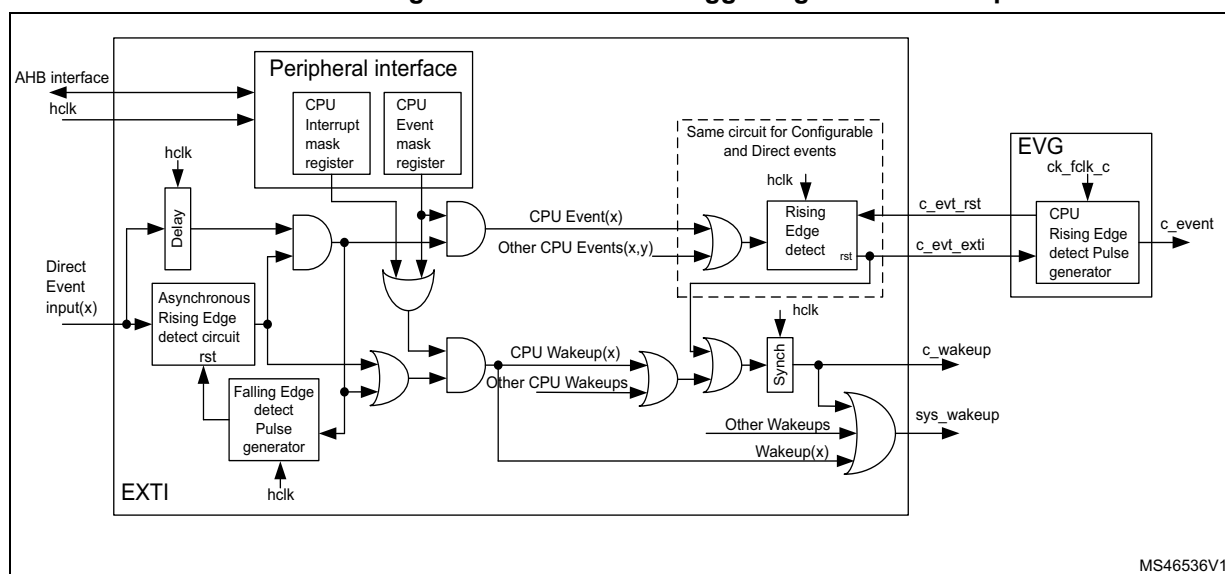
### 13.3.2 EXTI direct event input wake-up

Figure 28 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 wake-up event. The peripheral synchronous interrupt, associated with the direct wake-up 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.

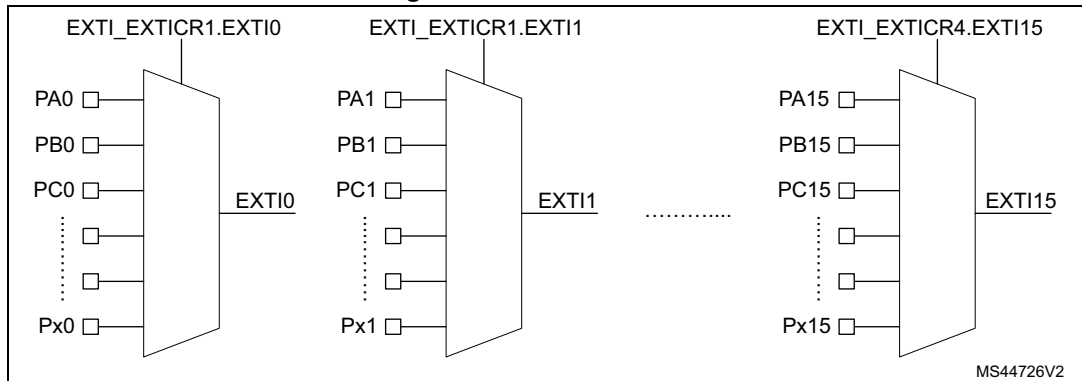
Figure 28. Direct event trigger logic CPU wake-up



### 13.3.3 EXTI mux

The EXTI mux allows selecting GPIOs as interrupts and wake-up. The GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI mux output is controlled through the [EXTI external interrupt selection register \(EXTI\\_EXTICRx\)](#) register.

Figure 29. EXTI GPIO mux



The EXTI mux outputs are available as output signals from the EXTI, to trigger other functional blocks. The EXTI mux 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 65. EXTI line connections

EXTI line	Line source	Line type
0-15	GPIO	Configurable
16	PVD output	Configurable
17	COMP1 output	Configurable
18	COMP2 output	Configurable
19	RTC	Direct
20	COMP3 output	Configurable
21	TAMP	Direct
22	I2C2 wake-up	Direct
23	I2C1 wake-up	Direct
24	USART3 wake-up	Direct
25	USART1 wake-up	Direct
26	USART2 wake-up	Direct
27	CEC wake-up	Direct
28	LPUART1 wake-up	Direct
29	LPTIM1	Direct
30	LPTIM2	Direct
31	LSE_CSS	Direct
32	UCPD1 wake-up	Direct
33	UCPD2 wake-up	Direct
34	V <sub>DDIO2</sub> monitoring	Configurable

Table 65. EXTI line connections (continued)

EXTI line	Line source	Line type
35	LPUART2 wake-up	Direct
36	USB wake-up	Direct

## 13.4 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wake-up 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 wake-up is conditioned by the CPU interrupt mask and CPU event mask.

Table 66. 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 wake-up
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\_RTISR1 register setting.

### 13.5 EXTI registers

The EXTI register map is divided in the following sections:

**Table 67. 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.

#### 13.5.1 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT20	Res.	RT18	RT17	RT16
											r/w		r/w	r/w	r/w
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
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **RT20**: Rising trigger event configuration bit of configurable line 20<sup>(1)</sup>

This bit enables/disables the rising edge trigger for the event and interrupt on the corresponding line.

0: Disable

1: Enable

The RT20 bit is only available in STM32G0B1xx and STM32G0C1xx. It is reserved in all the other devices.

Bit 19 Reserved, must be kept at reset value.

Bits 18:0 **RTx**: Rising trigger event configuration bit of configurable line x (x = 18 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 RT18 and RT17 bits are reserved in STM32G031xx and STM32G041xx.

1. The configurable lines are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable line occurs during writing of the register, the associated pending bit is not set. Rising edge trigger can be set for a line with falling edge trigger enabled. In this case, both edges generate a trigger.

### 13.5.2 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT20	Res.	FT18	FT17	FT16
											rw		rw	rw	rw
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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FT20**: Falling trigger event configuration bit of configurable line 20<sup>(1)</sup>.

This bit enables/disables the falling edge trigger for the event and interrupt on the corresponding line.

- 0: Disable
- 1: Enable

The FT20 bit is only available in STM32G0B1xx and STM32G0C1xx. It is reserved in all the other devices.

Bit 19 Reserved, must be kept at reset value.

Bits 18:0 **FTx**: Falling trigger event configuration bit of configurable line x (x = 18 to 0)<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 FT18 and FT17 bits are reserved in STM32G031xx and STM32G041xx.

1. The configurable lines are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable line occurs during writing of the register, the associated pending bit is not set. Falling edge trigger can be set for a line with rising edge trigger enabled. In this case, both edges generate a trigger.

### 13.5.3 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.	SWI 20	Res.	SWI 18	SWI 17	SWI 16
											rw		rw	rw	rw
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 31:21 Reserved, must be kept at reset value.

Bit 20 **SWI20**: Software rising edge event trigger on line 20.

Setting this bit by software triggers a rising edge event on the corresponding line, resulting in an interrupt, independently of EXTI\_RTSR1 and EXTI\_FTSR1 settings. This bit is automatically cleared by hardware. Reading this bit always returns 0.

0: No effect

1: Rising edge event generated on the corresponding line, followed by an interrupt

The SWI20 bit is only available in STM32G0B1xx and STM32G0C1xx. It is reserved in all the other devices.

Bit 19 Reserved, must be kept at reset value.

Bits 18:0 **SWIx**: Software rising edge event trigger on line x (x = 18 to 0)

Setting of any bit by software triggers a rising edge event on the corresponding line x, resulting in an interrupt, independently of EXTI\_RTSR1 and EXTI\_FTSR1 settings. These bits are automatically cleared by HW. Reading of any bit always returns 0.

0: No effect

1: Rising edge event generated on the corresponding line, followed by an interrupt

The SWI18 and SWI17 bits are reserved in STM32G031xx and STM32G041xx.

### 13.5.4 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.	RPIF20	Res.	RPIF18	RPIF17	RPIF16
											rc_w1		rc_w1	rc_w1	rc_w1
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 31:21 Reserved, must be kept at reset value.

Bit 20 **RPIF20**: Rising edge event pending for configurable line 20.

This bit is set upon a rising edge event generated by hardware or by software (through the EXTI\_SWIER1 register) on the corresponding line. This bit is cleared by writing 1 into it.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

The RPIF20 bit is only available in STM32G0B1xx and STM32G0C1xx. It is reserved in all the other devices.

Bit 19 Reserved, must be kept at reset value.

Bits 18:0 **RPIF<sub>x</sub>**: Rising edge event pending for configurable line x (x = 18 to 0)

Each bit is set upon a rising edge event generated by hardware or by software (through the EXTI\_SWIER1 register) on the corresponding line. Each bit is cleared by writing 1 into it.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

The RPIF18 and RPIF17 bits are reserved in STM32G031xx and STM32G041xx.



### 13.5.5 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.	FPIF20	Res.	FPIF18	FPIF17	FPIF16
											rc_w1		rc_w1	rc_w1	rc_w1
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 31:21 Reserved, must be kept at reset value.

Bit 20 **FPIF20**: Falling edge event pending for configurable line 20.

This bit is set upon a falling edge event generated by hardware on the corresponding line.  
This bit is cleared by writing 1 into it.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

The FPIF20 bit is only available in STM32G0B1xx and STM32G0C1xx. It is reserved in all the other devices.

Bit 19 Reserved, must be kept at reset value.

Bits 18:0 **FPIF<sub>x</sub>**: Falling edge event pending for configurable line x (x = 18 to 0)

Each bit is set upon a falling edge event generated by hardware on the corresponding line.  
Each bit is cleared by writing 1 into it.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

The FPIF18 and FPIF17 bits are reserved in STM32G031xx and STM32G041xx.

### 13.5.6 EXTI rising trigger selection register 2 (EXTI\_RTSR2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

This register is not available in STM32G031xx and STM32G041xx, and in STM32G051xx and STM32G061xx.

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.	RT34	Res.	Res.
													rw		

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

Bit 2 **RT34**: Rising trigger event configuration bit of configurable line 34<sup>(1)</sup>

This bit enables/disables the rising edge trigger for the event and interrupt on line 34.

0: Disable

1: Enable

This bit is only available in STM32G0B1xx and STM32G0C1xx. Reserved in all the other devices.

Bits 1:0 Reserved, must be kept at reset value.

- 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.

### 13.5.7 EXTI falling trigger selection register 2 (EXTI\_FTSR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

This register is not available in STM32G031xx and STM32G041xx, and in STM32G051xx and STM32G061xx.

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.	FT34	Res.	Res.
													rw		

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

Bit 2 **FT34**: Falling trigger event configuration bit of configurable line 34<sup>(1)</sup>.

This bit enables/disables the falling edge trigger for the event and interrupt on the corresponding line.

0: Disable

1: Enable

This bit is only available in STM32G0B1xx and STM32G0C1xx. Reserved in all the other devices.

Bits 1:0 Reserved, must be kept at reset value.

- 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.

### 13.5.8 EXTI software interrupt event register 2 (EXTI\_SWIER2)

Address offset: 0x030

Reset value: 0x0000 0000

Contains only register bits for configurable events.

This register is not available in STM32G031xx and STM32G041xx, and in STM32G051xx and STM32G061xx.

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.	SWI34	Res.	Res.
													rw		

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

Bit 2 **SWI34**: Software rising edge event trigger on line 34

Setting this bit by software triggers a rising edge event on the line 34, resulting in an interrupt, independently of EXTI\_RTSR2 and EXTI\_FTSR2 settings. This bit is automatically cleared by hardware. Reading this bit always returns 0.

0: No effect

1: Rising edge event generated on the line 34, followed by an interrupt

This bit is only available in STM32G0B1xx and STM32G0C1xx. Reserved in all the other devices.

Bits 1:0 Reserved, must be kept at reset value.

### 13.5.9 EXTI rising edge pending register 2 (EXTI\_RPR2)

Address offset: 0x034

Reset value: 0x0000 0000

Contains only register bits for configurable events.

This register is not available in STM32G031xx and STM32G041xx, and in STM32G051xx and STM32G061xx.

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.	RPIF34	Res.	Res.
													rc_w1		

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

Bit 2 **RPIF34**: Rising edge event pending for configurable line 34

This bit is set upon a rising edge event generated by hardware or by software (through the EXTI\_SWIER2 register) on line 34. This bit is cleared by writing 1 into it.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

This bit is only available in STM32G0B1xx and STM32G0C1xx. Reserved in all the other devices.

Bits 1:0 Reserved, must be kept at reset value.

### 13.5.10 EXTI falling edge pending register 2 (EXTI\_FPR2)

Address offset: 0x038



Reset value: 0x0000 0000

Contains only register bits for configurable events.

This register is not available in STM32G031xx and STM32G041xx, and in STM32G051xx and STM32G061xx.

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.	FPIF34	Res.	Res.
													rc_w1		

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

Bit 2 **FPIF34**: Falling edge event pending for configurable line 34

This bit is set upon a falling edge event generated by hardware on the line 34. This bit is cleared by writing 1 into it.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

This bit is only available in STM32G0B1xx and STM32G0C1xx. Reserved in all the other devices.

Bits 1:0 Reserved, must be kept at reset value.

### 13.5.11 EXTI external interrupt selection register (EXTI\_EXTICRx)

Address offset: 0x060 + 0x4 \* (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

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: PE[m+3] pin

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: PE[m+2] pin
- 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: PE[m+1] pin
- 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: PE[m] pin
- 0x05: PF[m] pin
- Other: Reserved

### 13.5.12 EXTI CPU wake-up with interrupt mask register (EXTI\_IMR1)

Address offset: 0x080

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	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IMx**: CPU wake-up with interrupt mask on line x (x = 31 to 0)

Setting/clearing each bit unmask/masks the CPU wake-up with interrupt, by an event on the corresponding line.

- 0: wake-up with interrupt masked
- 1: wake-up with interrupt unmasked

The IM24, IM22 and IM20 bits are only available in STM32G0B1xx and STM32G0C1xx. They are reserved in all the other devices.

The IM27 bit is only available in STM32G071xx and STM32G081xx, and in STM32G0B1xx and STM32G0C1xx. It is reserved in all the other devices.

The IM18 and IM17 bits are reserved in STM32G031xx and STM32G041xx.

### 13.5.13 EXTI CPU wake-up with event mask register (EXTI\_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
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
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **EMx**: CPU wake-up with event generation mask on line x (x = 31 to 0)

Setting/clearing each bit unmask/masks the CPU wake-up with event generation on the corresponding line.

- 0: wake-up with event generation masked
- 1: wake-up with event generation unmasked

The EM24, EM22 and EM20 bits are only available in STM32G0B1xx and STM32G0C1xx. They are reserved in all other devices.

The EM27 bit is only available in STM32G071xx and STM32G081xx, and STM32G0B1xx and STM32G0C1xx. It is reserved in all the other devices.

The EM18 and EM17 bits are reserved in STM32G031xx and STM32G041xx.

### 13.5.14 EXTI CPU wake-up with interrupt mask register (EXTI\_IMR2)

Address offset: 0x090 (EXTI\_IMR2)

Reset value: 0x0x0000 001B

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.

This register is not available in STM32G031xx and STM32G041xx, and in STM32G051xx and STM32G061xx.

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.	IM36	IM35	IM34	IM33	IM32
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **IMx**: CPU wake-up with interrupt mask on line x (x = 36 to 32)

Setting/clearing this bit unmask/masks the CPU wake-up with interrupt, by an event on the corresponding line.

0: wake-up with interrupt request from Line x is masked

1: wake-up with interrupt request from Line x is unmasked

The IM36, IM35, and IM34 bits are only available in STM32G0B1xx and STM32G0C1xx. They are reserved in STM32G071xx and STM32G081xx.

### 13.5.15 EXTI CPU wake-up with event mask register (EXTI\_EMR2)

Address offset: 0x094

Reset value: 0x0000 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.

This register is not available in STM32G031xx and STM32G041xx, and in STM32G051xx and STM32G061xx.

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.	EM36	EM35	EM34	EM33	EM32
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **EMx**: CPU wake-up with event generation mask on line x (x = 36 to 32)

Setting/clearing each bit unmask/masks the CPU wake-up with event generation on the corresponding line.

0: wake-up with event generation masked

1: wake-up with event generation unmasked

The EM36, EM35, and EM34 bits are only available in STM32G0B1xx and STM32G0C1xx. They are reserved in STM32G071xx and STM32G081xx.

### 13.5.16 EXTI register map

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

**Table 68. 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	EXTI_RTISR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT20	RT[18:0]														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
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT20	FT[18:0]														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
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI20	SWI[18:0]														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
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF20	RPIF[18:0]														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
0x010	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF20	FPIF[16:0] 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
0x014-0x024	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.	Res.
0x028	EXTI_RTISR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT34	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
0x02C	EXTI_FTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT34	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
0x030	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI34	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
0x034	EXTI_RPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF34	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
0x038	EXTI_FPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF34	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
0x038-0x05C	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.	Res.
0x060	EXTI_EXTICR1	EXTI13[7:0]							EXTI12[7:0]							EXTI11[7:0]							EXTI10[7:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	EXTI_EXTICR2	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	0	0
0x068	EXTI_EXTICR3	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	0	0







## 14 Cyclic redundancy check calculation unit (CRC)

### 14.1 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.

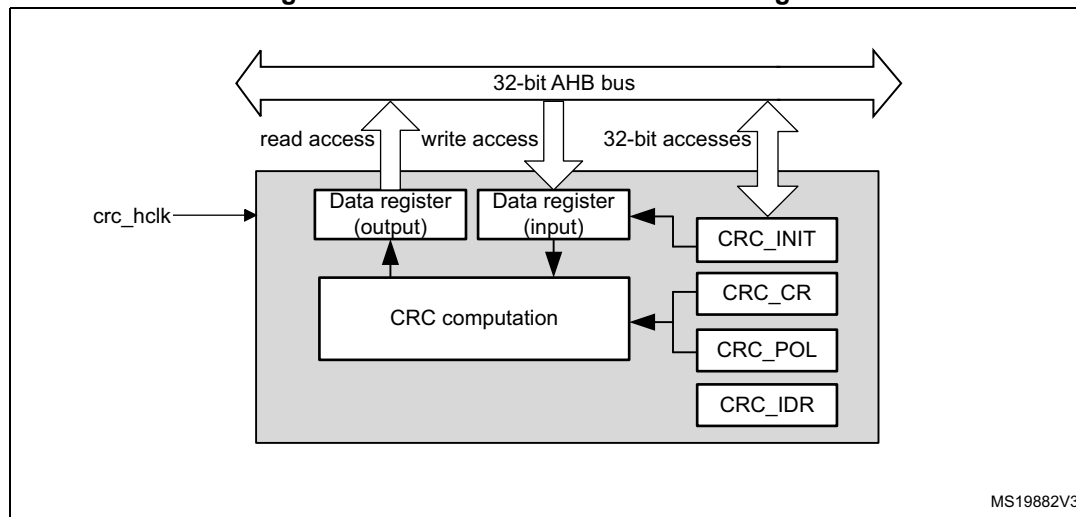
### 14.2 CRC main features

- 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

## 14.3 CRC functional description

### 14.3.1 CRC block diagram

Figure 30. CRC calculation unit block diagram



### 14.3.2 CRC internal signals

Table 69. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

### 14.3.3 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  $X+X^2+..+X^n$ , while the type of an odd polynomial is  $1+X+X^2+..+X^n$ .

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.

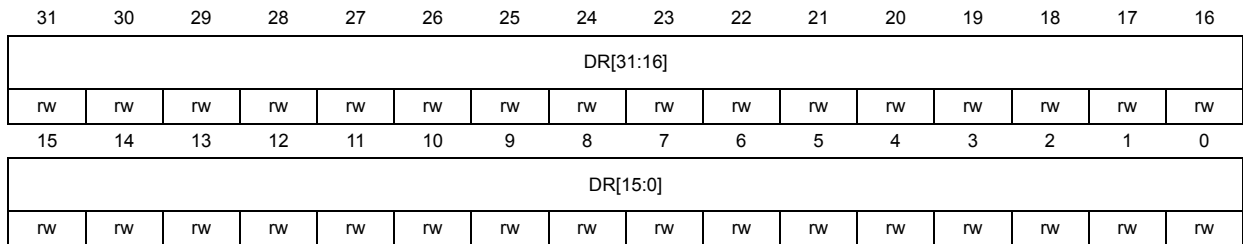
## 14.4 CRC registers

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.

### 14.4.1 CRC data register (CRC\_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF



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

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

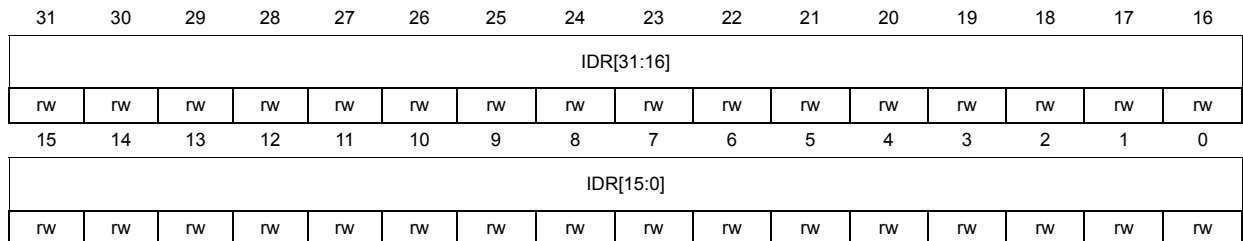
It holds the previous CRC calculation result when it is read.

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

### 14.4.2 CRC independent data register (CRC\_IDR)

Address offset: 0x04

Reset value: 0x0000 0000



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

### 14.4.3 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.	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.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept 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

### 14.4.4 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.

### 14.4.5 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.

14.4.6 CRC register map

Table 70. 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	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	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
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	1	0	1	1	0	1	1	0	1	1

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 15 Analog-to-digital converter (ADC)

### 15.1 Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 19 multiplexed channels allowing it to measure signals from 16 external and 3 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

A built-in hardware oversampler allows analog performances to be improved while off-loading the related computational burden from the CPU.

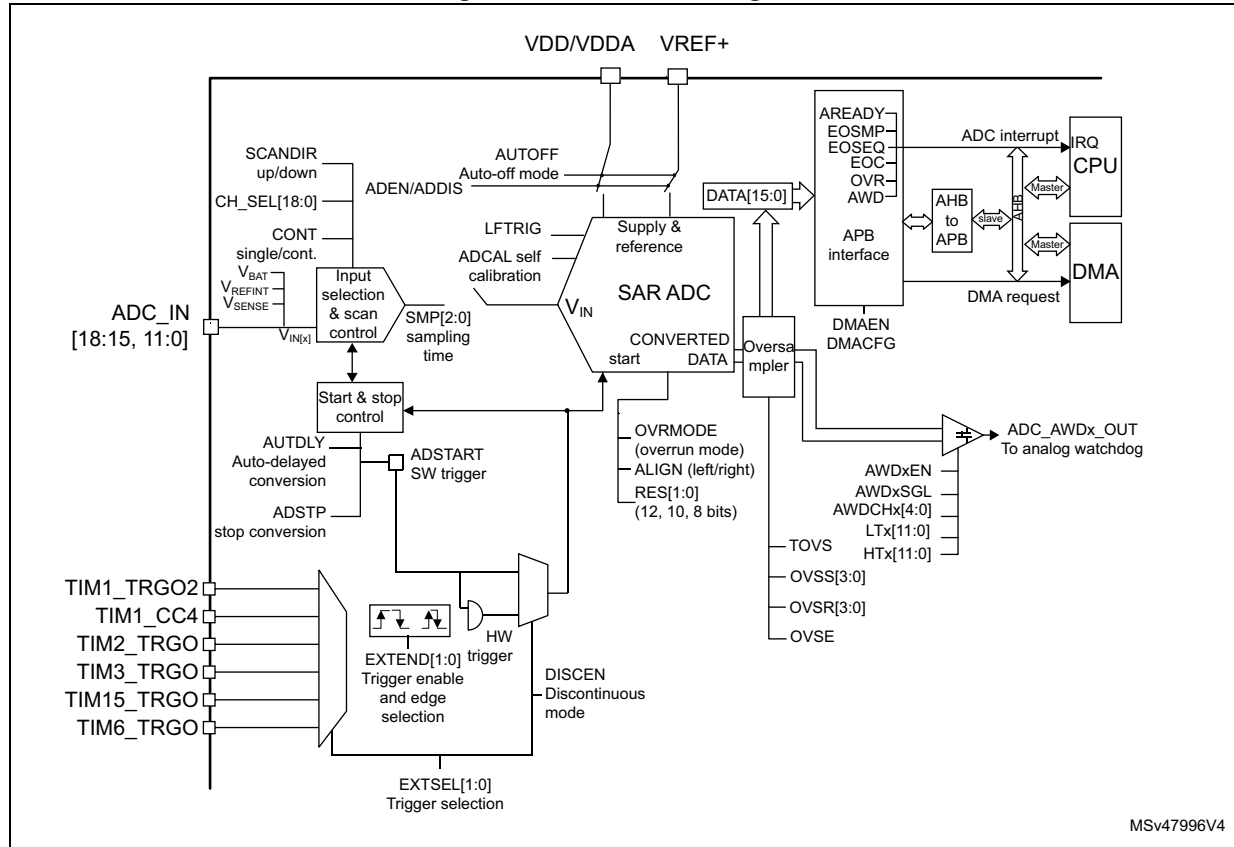
## 15.2 ADC main features

- 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
  - 16 external analog inputs
  - 1 channel for internal temperature sensor ( $V_{SENSE}$ )
  - 1 channel for internal reference voltage ( $V_{REFINT}$ )
  - 1 channel for monitoring external  $V_{BAT}$  power supply pin
- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (timer events or GPIO input events)
- Conversion modes
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events
- Analog watchdog
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8-bits
- ADC input range:  $V_{SSA} \leq V_{IN} \leq V_{REF+}$

### 15.3 ADC functional description

Figure 31 shows the ADC block diagram and Table 71 gives the ADC pin description.

Figure 31. ADC block diagram



#### 15.3.1 ADC pins and internal signals

Table 71. ADC input/output pins

Name	Signal type	Remarks
VDDA	Input, analog power supply	Analog power supply and positive reference voltage for the ADC
VSSA	Input, analog supply ground	Ground for analog power supply
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
ADC_INx	Analog input signals	16 external analog input channels

**Table 72. ADC internal input/output signals**

Internal signal name	Signal type	Description
V <sub>IN</sub> [x]	Analog input channels	Connected either to internal channels or to ADC_INi external channels
TRGx	Input	ADC conversion triggers
V <sub>SENSE</sub>	Input	Internal temperature sensor output voltage
V <sub>REFINT</sub>	Input	Internal voltage reference output voltage
V <sub>BAT/3</sub>	Input	VBAT pin input voltage divided by 3
ADC_AWDx_OUT	Output	Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number = 1,2,3)

**Table 73. External triggers**

Name	Source	EXTSEL[2:0]
TRG0	TIM1_TRGO2	000
TRG1	TIM1_CC4	001
TRG2	TIM2_TRGO	010
TRG3	TIM3_TRGO	011
TRG4	TIM15_TRGO	100
TRG5	TIM6_TRGO	101
TRG6	TIM4_TRGO	110
TRG7	EXTI11	111

### 15.3.2 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<sub>ADCVREG\_STUP</sub>) before launching a calibration or enabling the ADC. This delay must be managed by software (for details on t<sub>ADCVREG\_STUP</sub>, 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.

## 15.3.3 Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

Calibration should be performed before starting A/D conversion. It removes the offset error which may vary from chip to chip due to process variation.

The calibration is initiated by software by setting bit ADCAL to 1. It can be initiated only when all the following conditions are met:

- the ADC voltage regulator is enabled (ADVREGEN = 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 ( $V_{REF+}$  changes are the main contributor to ADC offset variations and temperature change to a lesser extend), it is recommended to re-run a calibration cycle.

The calibration factor is lost in the following cases:

- The power supply is removed from the ADC (for example when the product enters Standby or VBAT mode)
- The ADC peripheral is reset.

The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby or VBAT mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0). Then, at the next start of conversion, the calibration factor 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**

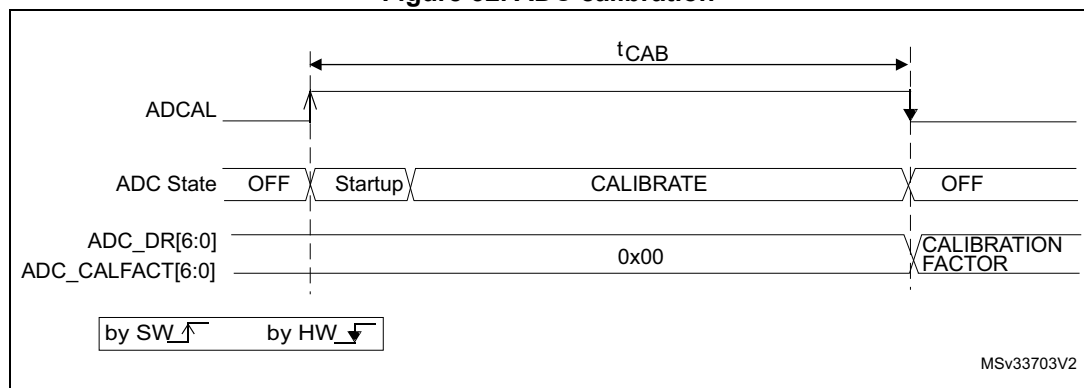
1. Ensure that ADEN = 0, AUTOFF = 0, ADVREGEN = 1, and DMAEN = 0.
2. Set ADCAL = 1.
3. Wait until ADCAL = 0 (or until EOCAL = 1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit in the ADC\_IER register.
4. The calibration factor minus one can then be read from bits 6:0 of the ADC\_DR or ADC\_CALFACT registers. The resulting calibration factor must be incremented by one and written back to the ADC\_CALFACT register.
5. To reduce the noise effect of the calibration factor extraction, the software can make the average of eight calibration factor values. This step is optional but recommended for better accuracy.

The averaging procedure is as follows

- a) Obtain eight incremented calibration factor values (perform eight times step 2 to step 4).
- b) Calculate the average of these eight values and round it up to the nearest integer.
- c) Write the result to the ADC\_CALFACT register.

*Note: If the resulting calibration factor is higher than 0x7F, write 0x7F to the ADC\_CALFACT register to avoid overflow.*

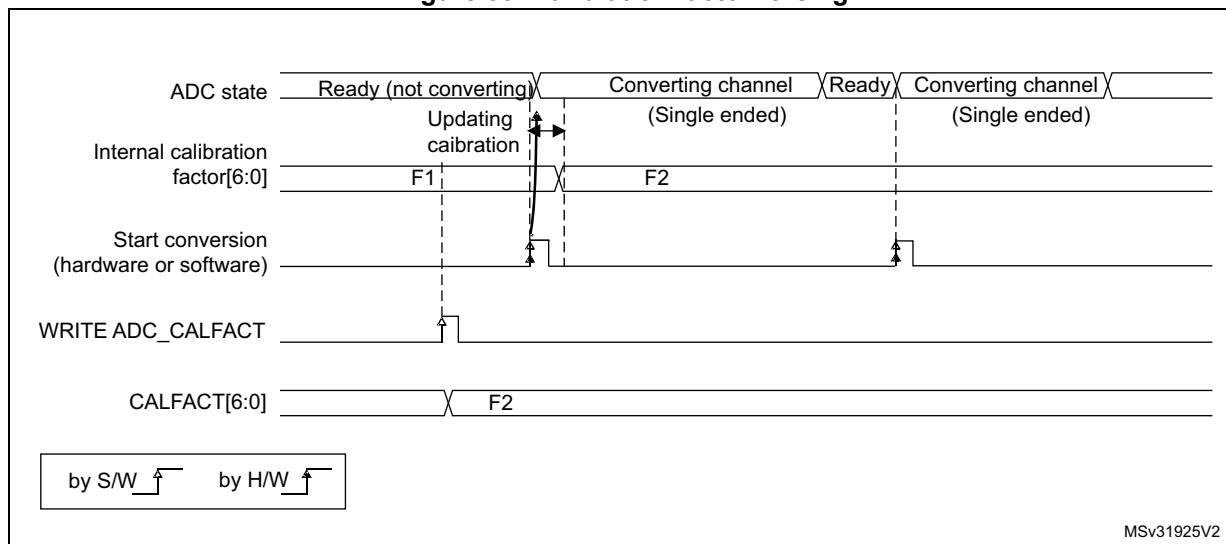
**Figure 32. ADC calibration**



### 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 33. Calibration factor forcing



### 15.3.4 ADC on-off control (ADEN, ADDIS, ADRDY)

At power-up, the ADC is disabled and put in power-down mode (ADEN = 0).

As shown in [Figure 34](#), 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 15.4: Conversion on external trigger and trigger polarity \(EXTSEL, EXTEN\)](#)) or when an external trigger event occurs if triggers are enabled.

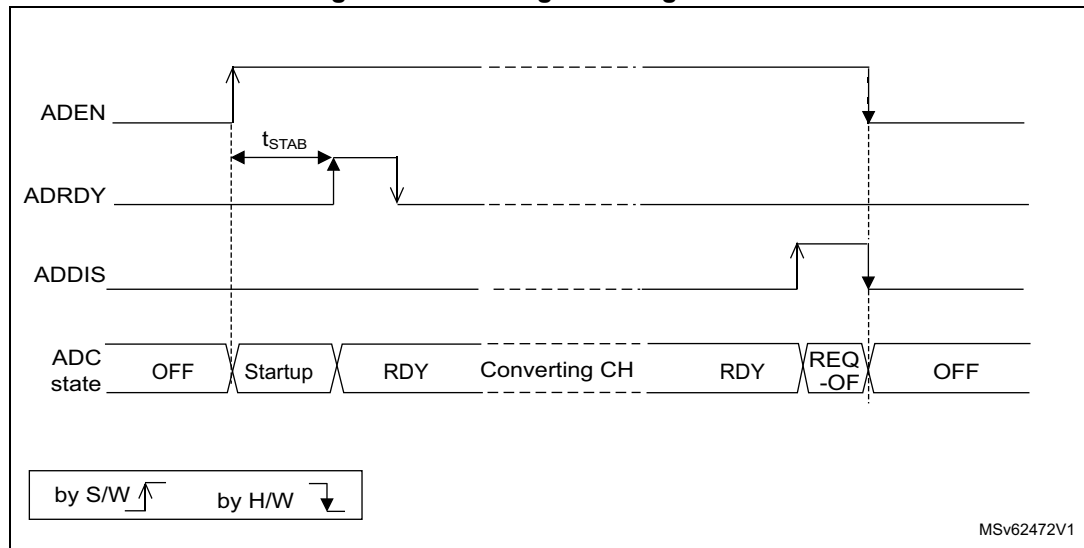
Follow this procedure to enable the ADC:

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 34. Enabling/disabling the ADC**



MSv62472V1

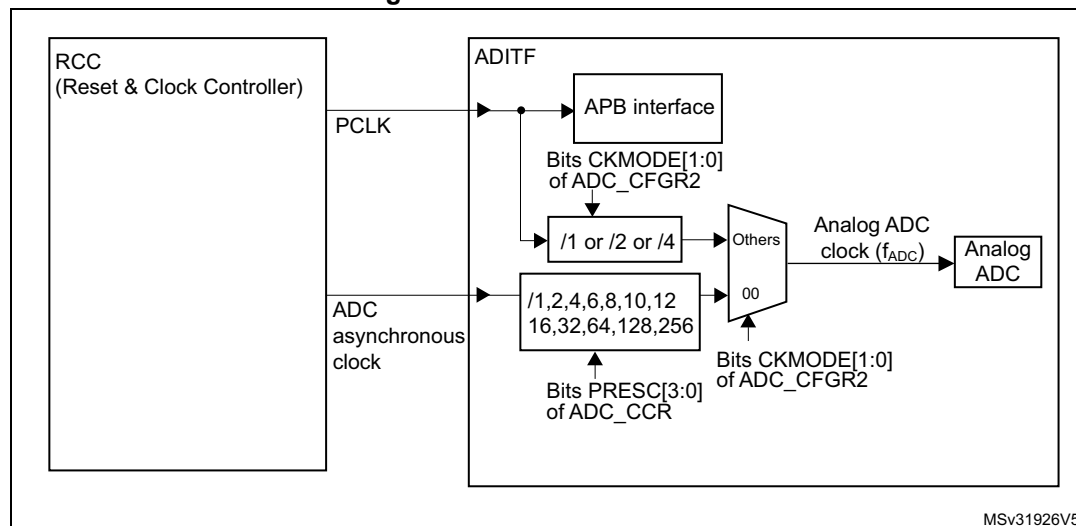
**Note:** In Auto-off mode ( $AUTOFF = 1$ ) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.  
 When the bus clock is much faster than the analog ADC clock ( $f_{ADC}$ ), a minimum delay of ten  $f_{ADC}$  clock cycles must be respected between ADEN and ADDIS bit settings.



### 15.3.5 ADC clock (CKMODE, PRESC[3:0])

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC asynchronous clock) independent from the APB clock (PCLK).

Figure 35. ADC clock scheme



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 35: 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 74. Latency between trigger and start of conversion<sup>(1)(2)</sup>

ADC clock source	CKMODE[1:0]	Latency between the trigger event and the start of conversion
HSI16, SYSCLK, or PLLPCLK <sup>(3)</sup>	00	Latency is not deterministic (jitter)
PCLK divided by 2	01	Latency is deterministic (no jitter) and equal to 3.25 $f_{ADC}$ cycles
PCLK divided by 4	10	Latency is deterministic (no jitter) and equal to 3.125 $f_{ADC}$ cycles
PCLK divided by 1	11	Latency is deterministic (no jitter) and equal to 3.5 $f_{ADC}$ cycles

1. Refer to the device datasheet for the maximum  $f_{ADC}$  frequency.
2. If the trigger is generated by TIM1 or TIM15 clocked at twice the CPU clock frequency, then the latency is not deterministic and can be increased by one TIM1 or TIM15 clock cycle.
3. Selected with ADCSEL bitfield of the RCC\_CCIPR register.

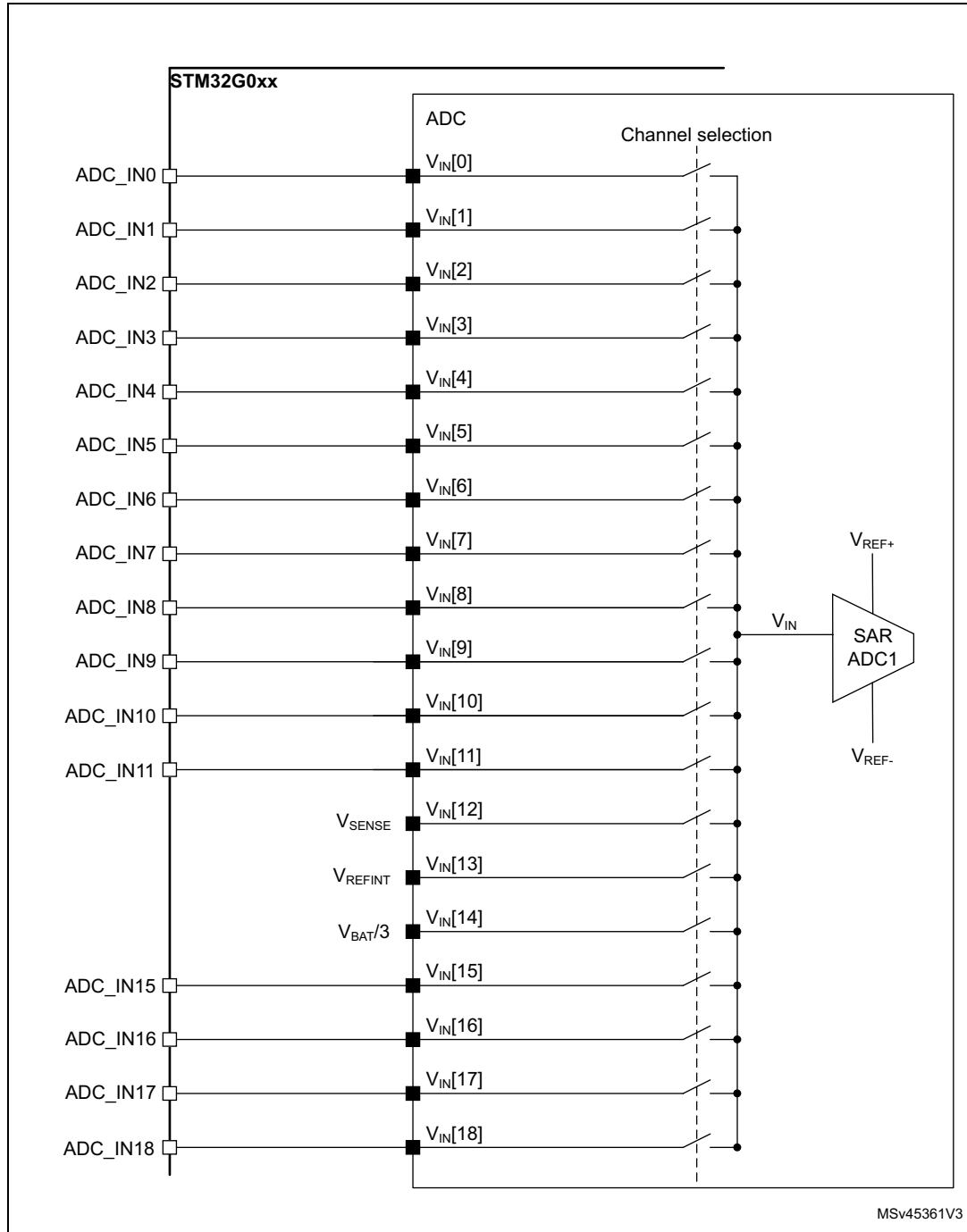
**Caution:** For correct operation of the ADC analog block, the analog ADC clock ( $f_{ADC}$ ) must have a duty cycle ranging from 45% to 55%. This is granted when the incoming clock (PCLK or ADC asynchronous clock) is divided by a factor of two or higher, using one of the scaler blocks inside the ADC. If it is not the case, some additional rules must be followed:

- When CKMODE[1:0] = 11 (PCLK divided by one), the AHB and APB prescalers in the RCC must be configured in bypass mode.
- When the analog ADC clock is derived from the HSE or LSE bypass clock, this bypass clock must have a 45-to-55% duty cycle unless this clock is routed to the ADC through the PLL.

### 15.3.6 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in [Figure 36](#).

Figure 36. ADC connectivity



### 15.3.7 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 15.12: 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).

### 15.3.8 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 19 multiplexed channels:

- 16 analog inputs from GPIO pins (ADC\_INx)
- 3 internal analog inputs (temperature sensor, internal reference voltage, V<sub>BAT</sub> channel)

It is possible to convert a single channel or a sequence of channels.

The sequence of the channels to be converted can be programmed in the ADC\_CHSELR channel selection register: each analog input channel has a dedicated selection bit (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>BAT</sub> internal channels

The temperature sensor is connected to channel ADC V<sub>IN</sub>[12].

The internal voltage reference V<sub>REFINT</sub> is connected to channel ADC V<sub>IN</sub>[13].

V<sub>BAT</sub> channel is connected to ADC V<sub>IN</sub>[14] channel.

When V<sub>REF+</sub> is lower than V<sub>DDA</sub>, this channel is not converted.

### 15.3.9 Programmable sampling time (SMPx[2:0])

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows 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  $f_{\text{ADC}} = 16 \text{ MHz}$  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.

### 15.3.10 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT = 0 in the ADC\_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC\_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC\_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

*Note:* To convert a single channel, program a sequence with a length of 1.

### 15.3.11 Continuous conversion mode (CONT = 1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT = 1 in the ADC\_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC\_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC\_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

*Note:* To convert a single channel, program a sequence with a length of 1.

*It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.*

### 15.3.12 Starting conversions (ADSTART)

Software starts ADC conversions by setting ADSTART = 1.

When ADSTART is set, the conversion:

- Starts immediately if EXTEN = 00 (software trigger)
- At the next active edge of the selected hardware trigger if EXTEN ≠ 00

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:

- In single mode with software trigger (CONT = 0, EXTEN = 00)
  - At any end of conversion sequence (EOS = 1)
- In discontinuous mode with software trigger (CONT = 0, DISCEN = 1, EXTEN = 00)
  - At end of conversion (EOC = 1)
- In all cases (CONT = x, EXTEN = XX)
  - After execution of the ADSTP procedure invoked by software (see [Section 15.3.14: Stopping an ongoing conversion \(ADSTP\) on page 357](#))

*Note:* In continuous mode (CONT = 1), the ADSTART bit is not cleared by hardware when the EOS flag is set because the sequence is automatically relaunched.

*When hardware trigger is selected in single mode (CONT = 0 and EXTEN = 01), ADSTART is not cleared by hardware when the EOS flag is set (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\_CHSELR register or changing CHSELRMOD or SCANDIR), it is mandatory to wait until CCRDY flag is asserted before asserting ADSTART, otherwise the value written to ADSTART is ignored.*

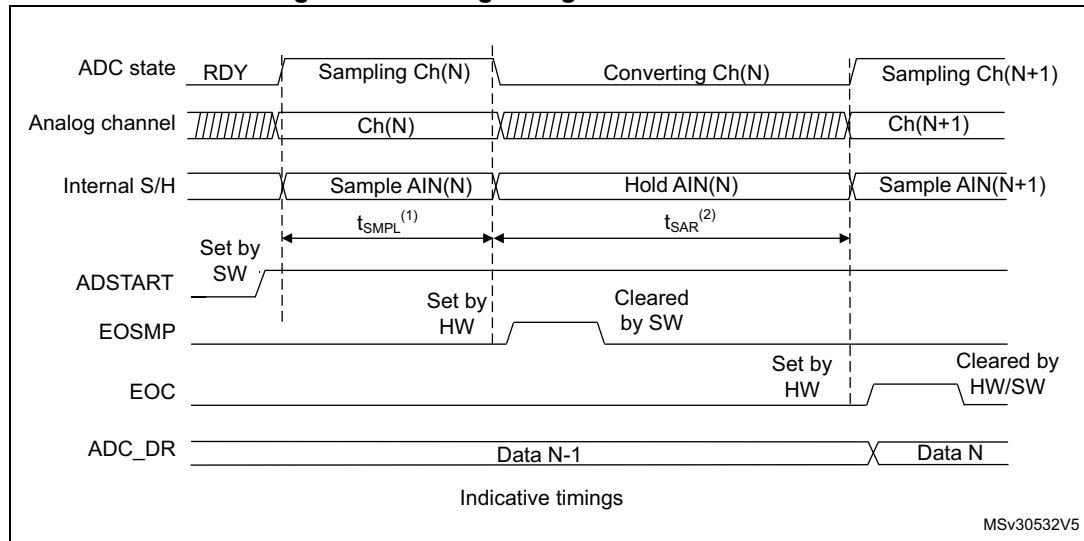
### 15.3.13 Timings

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$t_{CONV} = t_{SMPL} + t_{SAR} = [1.5_{|min} + 12.5_{|12bit}] \times 1/f_{ADC}$$

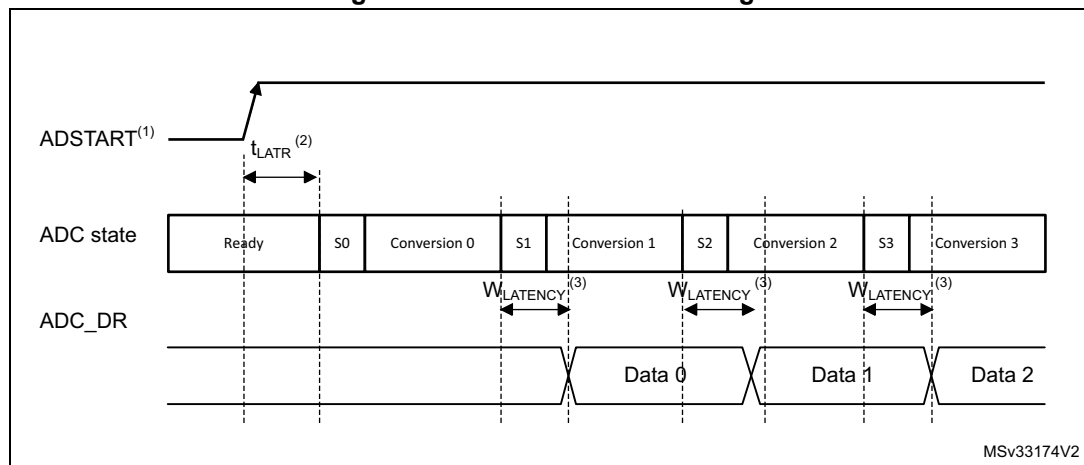
$$t_{CONV} = t_{SMPL} + t_{SAR} = 42.9 \text{ ns}_{|min} + 357.1 \text{ ns}_{|12bit} = 0.400 \text{ }\mu\text{s}_{|min} \text{ (for } f_{ADC} = 35 \text{ MHz)}$$

Figure 37. Analog-to-digital conversion time



1.  $t_{SMPL}$  depends on SMP[2:0].
2.  $t_{SAR}$  depends on RES[2:0].
3. The synchronization between the analog clock and the digital clock domains is not described in the above figure.

Figure 38. 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).



### 15.3.14 Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP = 1 in the ADC\_CR register.

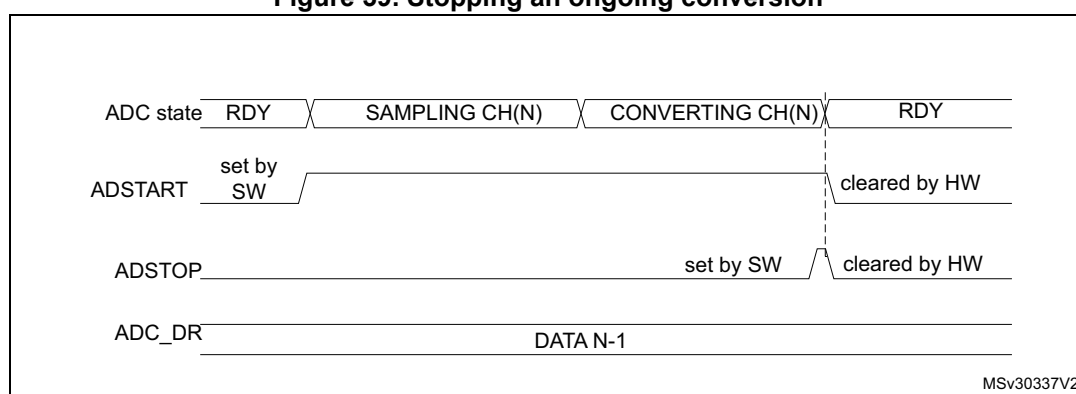
This resets the ADC operation and the ADC is idle, ready for a new operation.

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 39. Stopping an ongoing conversion**



## 15.4 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

If bit ADSTART = 0, any hardware triggers which occur are ignored.

[Table 75](#) provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

**Table 75. 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 73: External triggers](#) in [Section 15.3.1: 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 ( $ADSTART = 0$ ).

### 15.4.1 Discontinuous mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC\_CFGR1 register.

In this mode ( $DISCEN = 1$ ), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if  $DISCEN = 0$ , a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:

- $DISCEN = 1$ , channels to be converted = 0, 3, 7, 10
  - 1st trigger: channel 0 is converted and an EOC event is generated
  - 2nd trigger: channel 3 is converted and an EOC event is generated
  - 3rd trigger: channel 7 is converted and an EOC event is generated
  - 4th trigger: channel 10 is converted and both EOC and EOS events are generated.
  - 5th trigger: channel 0 is converted an EOC event is generated
  - 6th trigger: channel 3 is converted and an EOC event is generated
  - ...
- $DISCEN = 0$ , channels to be converted = 0, 3, 7, 10
  - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOS event.
  - Any subsequent trigger events 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  $CONT = 1$ .

### 15.4.2 Programmable resolution (RES) - Fast conversion mode

It is possible to obtain faster conversion times ( $t_{SAR}$ ) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC\_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

*Note:* The RES[1:0] bit must only be changed when the ADEN bit is reset.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeros.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in [Table 76](#).

**Table 76.  $t_{SAR}$  timings depending on resolution**

RES[1:0] (bits)	$t_{SAR}$ ( $f_{ADC}$ cycles)	$t_{SAR}$ at $f_{ADC} = 35$ MHz (ns)	$t_{SMPL(min)}$ ( $f_{ADC}$ cycles)	$t_{CONV}$ with min. $t_{SMPL}$ ( $f_{ADC}$ cycles)	$t_{CONV(min)}$ at $f_{ADC} = 35$ MHz (ns)
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

### 15.4.3 End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC\_ISR register as soon as a new conversion data result is available in the ADC\_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC\_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC\_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC\_ISR register. The EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC\_IER register.

The aim of this interrupt is to allow the processing to be synchronized with the conversions. Typically, an analog multiplexer can be accessed in hidden time during the conversion phase, so that the multiplexer is positioned when the next sampling starts.

*Note: As there is only a very short time left between the end of the sampling and the end of the conversion, it is recommended to use polling or a WFE instruction rather than an interrupt and a WFI instruction.*

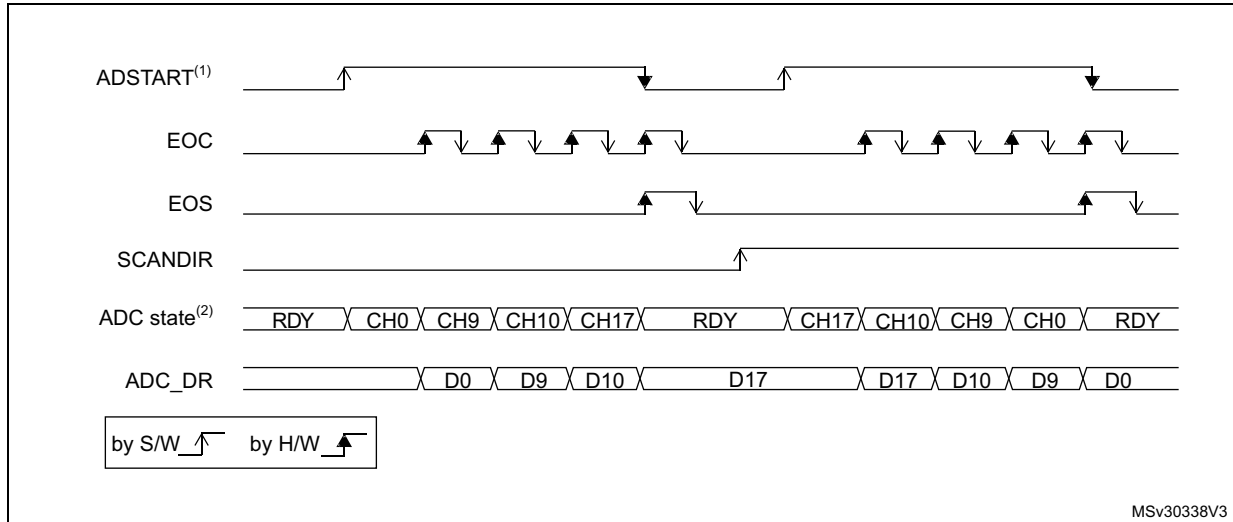
### 15.4.4 End of conversion sequence (EOS flag)

The ADC notifies the application of each end of sequence (EOS) event.

The ADC sets the EOS flag in the ADC\_ISR register as soon as the last data result of a conversion sequence is available in the ADC\_DR register. An interrupt can be generated if the EOSIE bit is set in the ADC\_IER register. The EOS flag is cleared by software by writing 1 to it.

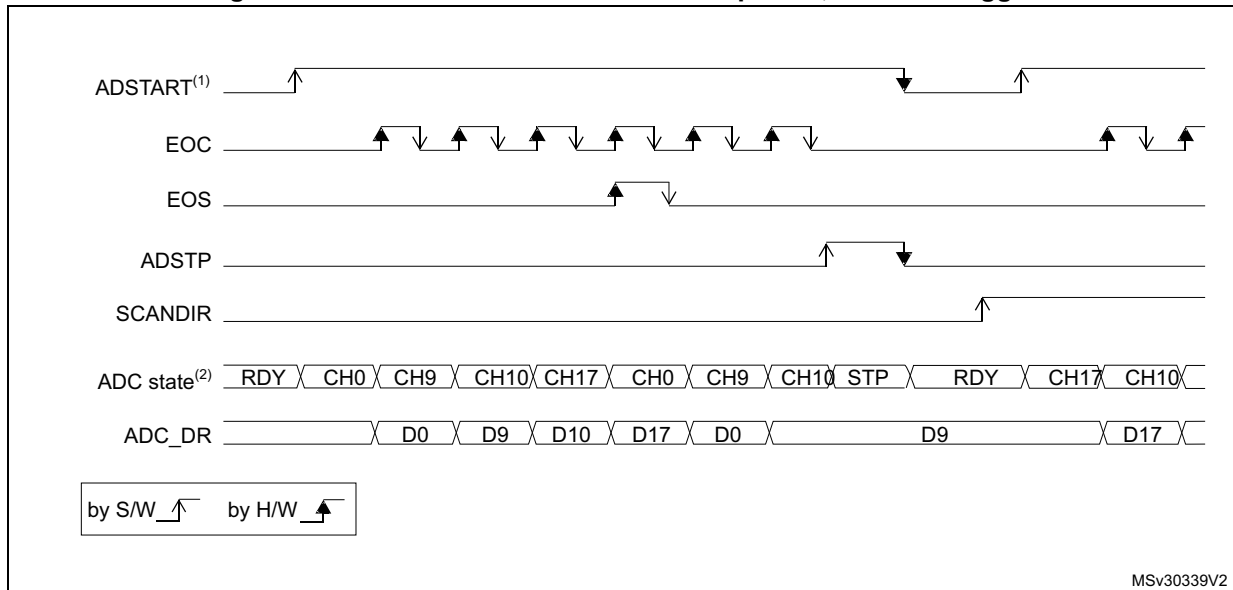
15.4.5 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 40. Single conversions of a sequence, software trigger



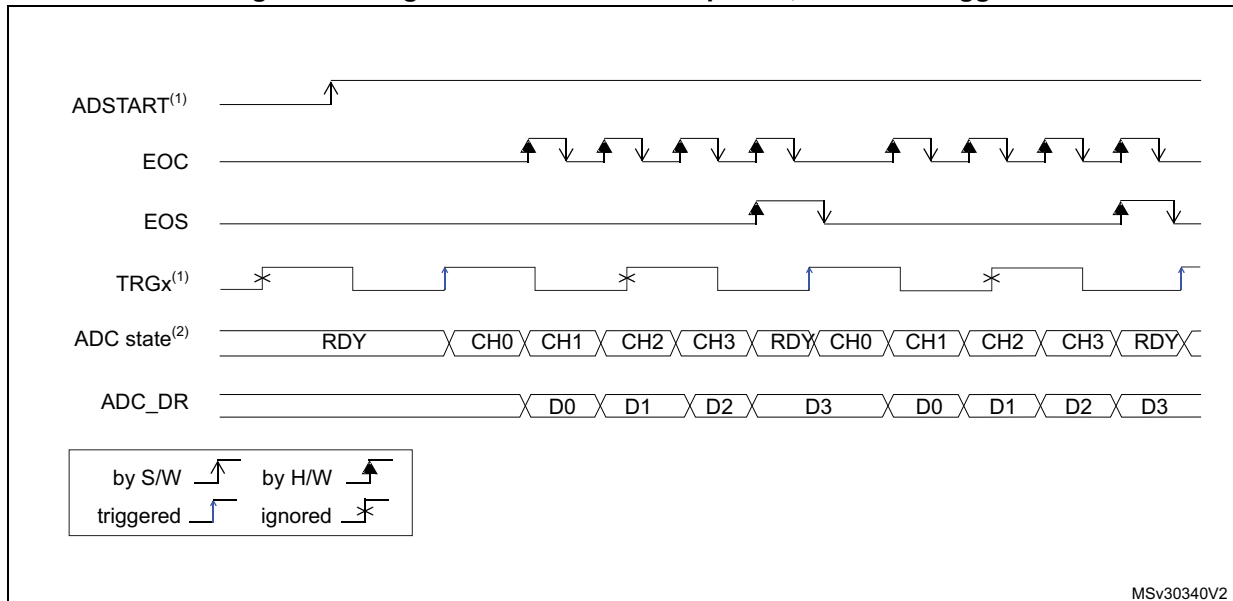
1. EXTEN = 00, CONT = 0
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

Figure 41. Continuous conversion of a sequence, software trigger



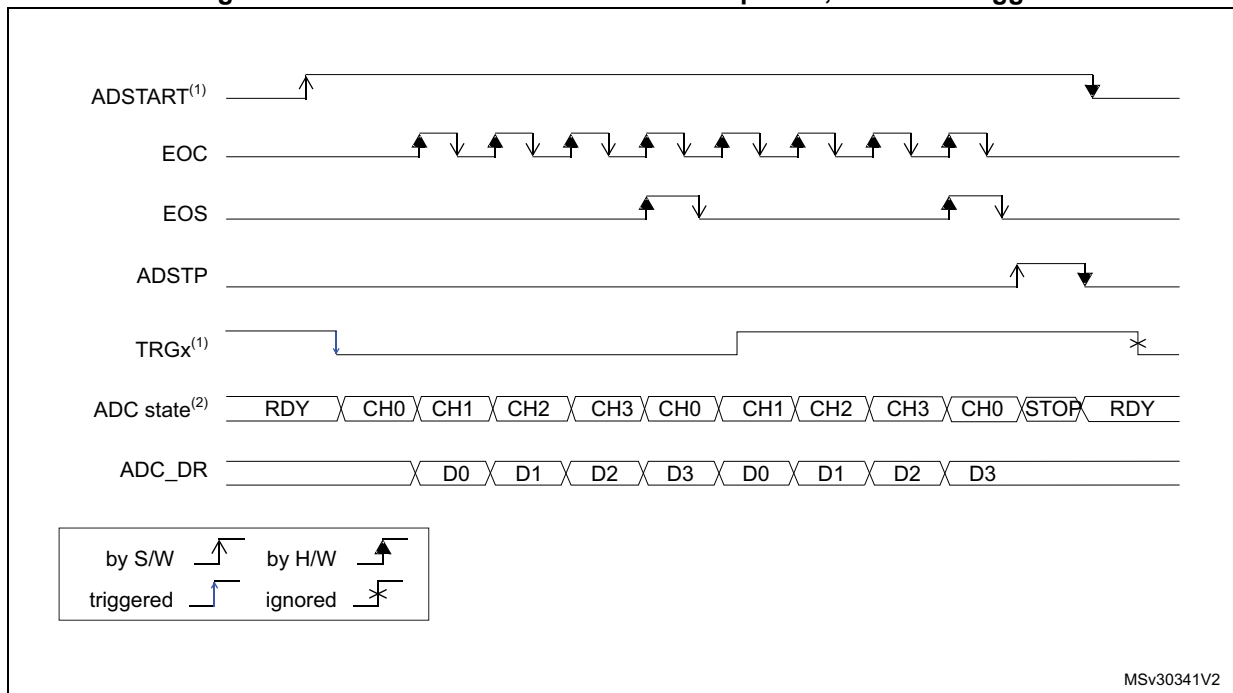
1. EXTEN = 00, CONT = 1,
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0

Figure 42. Single conversions of a sequence, hardware trigger



1. EXTSEL = TRGx (over-frequency), EXTEN = 01 (rising edge), CONT = 0
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

Figure 43. Continuous conversions of a sequence, hardware trigger



1. EXTSEL = TRGx, EXTEN = 10 (falling edge), CONT = 1
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0

### 15.4.6 Low frequency trigger mode

Once the ADC is enabled or the last ADC conversion is complete, the ADC is ready to start a new conversion. The ADC needs to be started at a predefined time ( $t_{idle}$ ) 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.

## 15.5 Data management

### 15.5.1 Data register and data alignment (ADC\_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC\_DR data register which is 16-bit wide.

The format of the ADC\_DR depends on the configured data alignment and resolution.

The ALIGN bit in the ADC\_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN = 0) or left-aligned (ALIGN = 1) as shown in [Figure 44](#).

**Figure 44. 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]											

MS30342V1

### 15.5.2 ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the data from a new conversion is available.

The OVR flag is set in the ADC\_ISR register if the EOC flag is still at '1' at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC\_IER register.

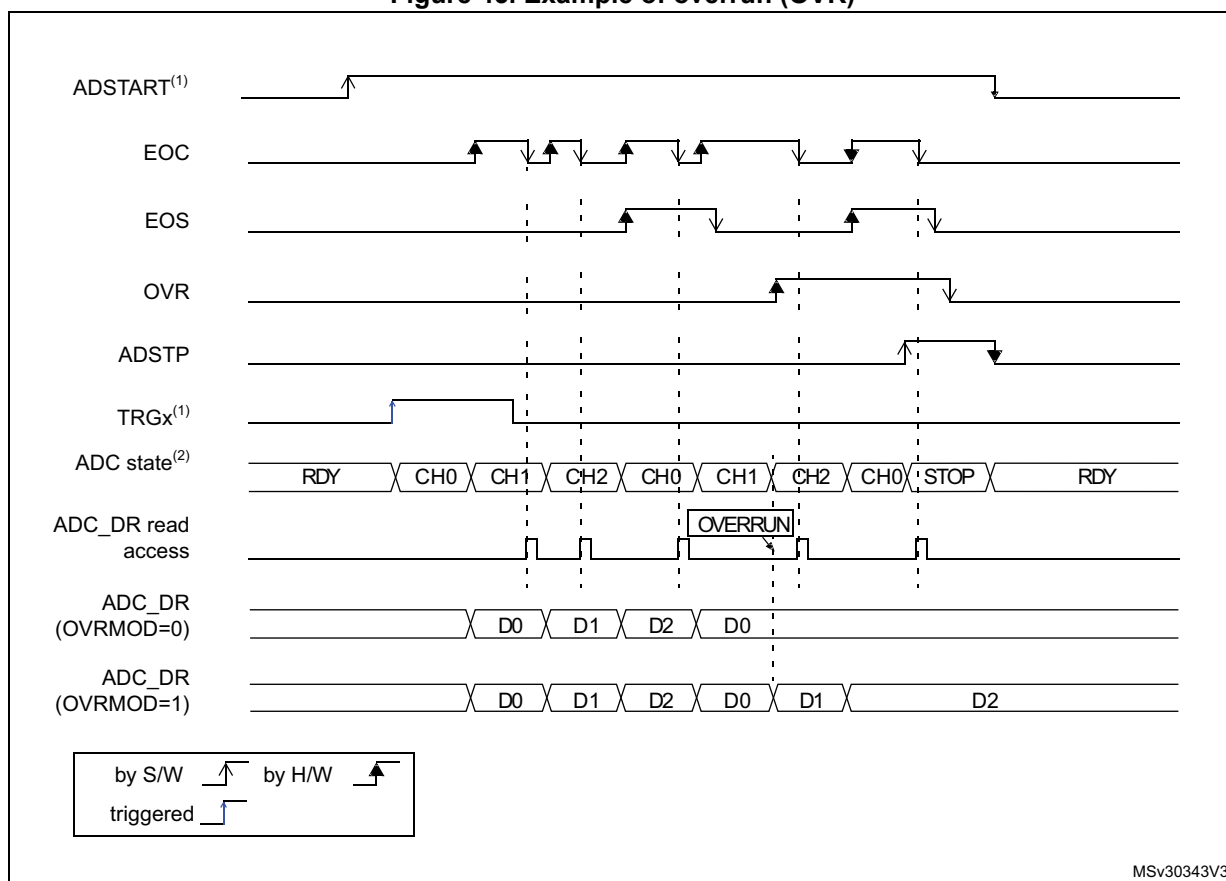
When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC\_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVRMOD bit in the ADC\_CFGR1 register:

- OVRMOD = 0
  - An overrun event preserves the data register from being overwritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is discarded.
- OVRMOD = 1
  - The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, further conversions can be performed and the ADC\_DR register always contains the data from the latest conversion.

Figure 45. Example of overrun (OVR)



### 15.5.3 Managing a sequence of data converted without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC\_ISR register and the ADC\_DR register can be read. The OVRMOD bit in the ADC\_CFGR1 register should be configured to 0 to manage overrun events as an error.

### 15.5.4 Managing converted data without using the DMA without overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag should be ignored by the software. When OVRMOD = 1, an overrun event does not prevent the ADC from continuing to convert and the ADC\_DR register always contains the latest conversion data.

### 15.5.5 Managing converted data using the DMA

Since all converted channel values are stored in a single data register, it is efficient to use DMA when converting more than one channel. This avoids losing the conversion data results stored in the ADC\_DR register.

When DMA mode is enabled (DMAEN bit set 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 15.5.2: ADC overrun \(OVR, OVRMOD\) on page 362](#)).

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 10: Direct memory access controller \(DMA\) on page 275](#)) 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.

## **15.6 Low-power features**

### **15.6.1 Wait mode conversion**

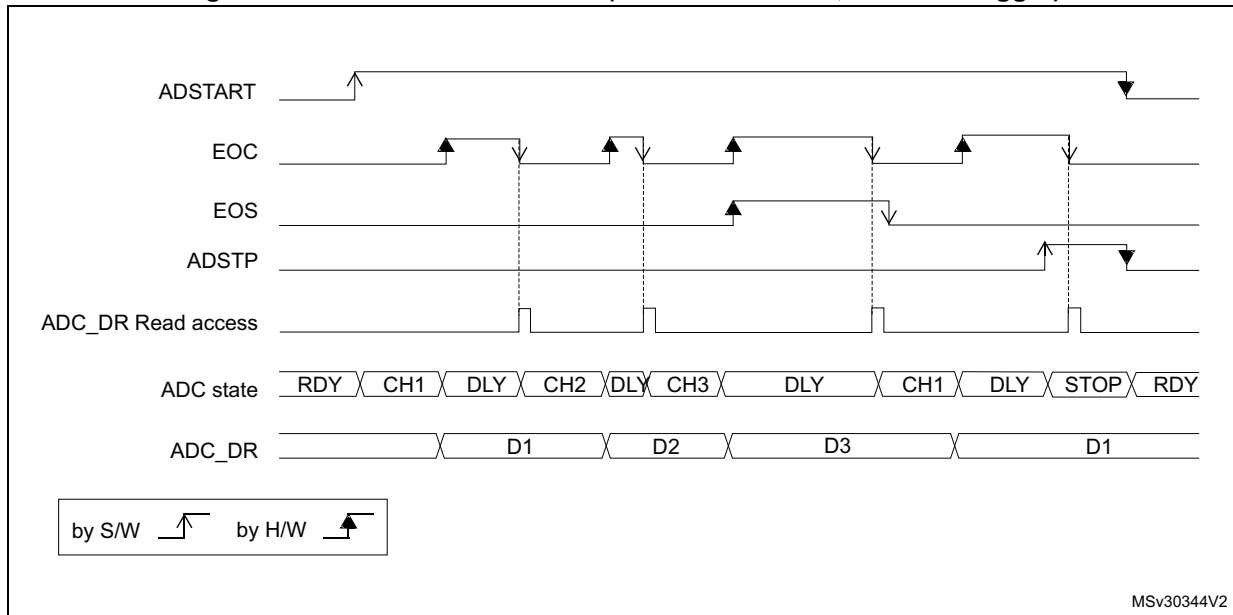
Wait mode conversion can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set 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 46. Wait mode conversion (continuous mode, software trigger)



1. EXTEN = 00, CONT = 1
2. CHSEL = 0x3, SCANDIR = 0, WAIT = 1, AUTOFF = 0

### 15.6.2 Auto-off mode (AUTOFF)

The ADC has an automatic power management feature which is called auto-off mode, and is enabled by setting AUTOFF = 1 in the ADC\_CFGR1 register.

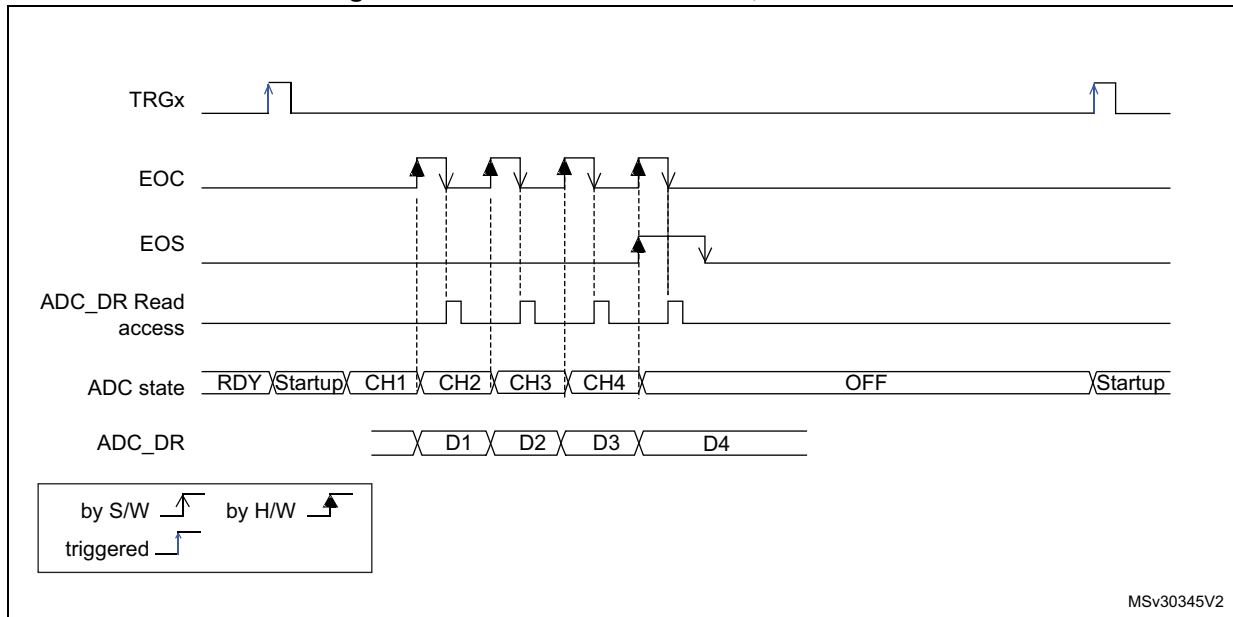
When AUTOFF = 1, the ADC is always powered off when not converting and automatically wakes-up when a conversion is started (by software or hardware trigger). A startup-time is automatically inserted between the trigger event which starts the conversion and the sampling time of the ADC. The ADC is then automatically disabled once the sequence of conversions is complete.

Auto-off mode can cause a dramatic reduction in the power consumption of applications which need relatively few conversions or when conversion requests are timed far enough apart (for example with a low frequency hardware trigger) to justify the extra power and extra time used for switching the ADC on and off.

Auto-off mode can be combined with the wait mode conversion (WAIT = 1) for applications clocked at low frequency. This combination can provide significant power savings if the ADC is automatically powered-off during the wait phase and restarted as soon as the ADC\_DR register is read by the application (see [Figure 48: Behavior with WAIT = 1, AUTOFF = 1](#)).

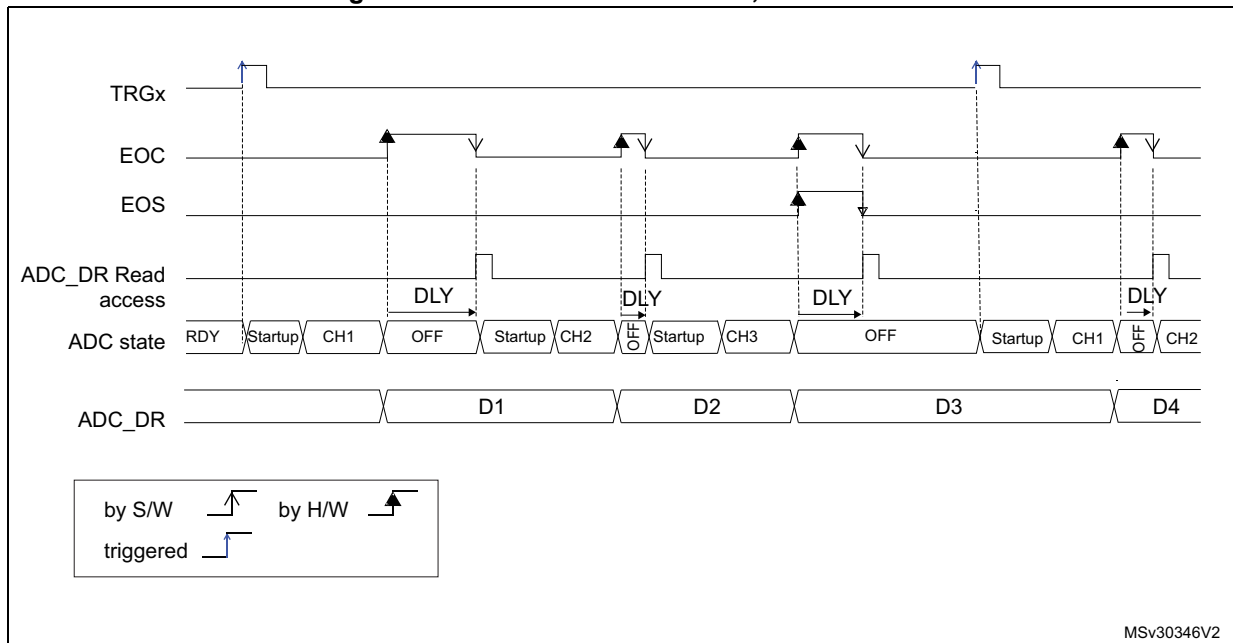
*Note:* Refer to the Section *Reset and clock control (RCC)* for the description of how to manage the dedicated 14 MHz internal oscillator. The ADC interface can automatically switch ON/OFF the 14 MHz internal oscillator to save power.

Figure 47. 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 48. Behavior with WAIT = 1, AUTOFF = 1



- EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1

## 15.7 Analog window watchdogs

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

### 15.7.1 Description of analog watchdog 1

AWD1 analog watchdog is enabled by setting the AWD1EN bit in the ADC\_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels (see [Table 78: Analog watchdog 1 channel selection](#)) remain within a configured voltage range (window) as shown in [Figure 49](#).

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in HT1[11:0] and LT1[11:0] bits of ADC\_AWD1TR register. An interrupt can be enabled by setting the AWD1IE bit in the ADC\_IER register.

The AWD1 flag is cleared by software by 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 77](#) describes how the comparison is performed for all the possible resolutions.

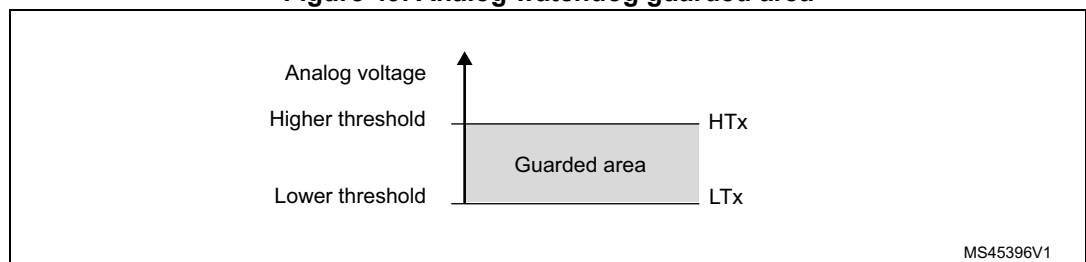
**Table 77. Analog watchdog comparison**

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 78](#) 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 49. Analog watchdog guarded area**



**Table 78. Analog watchdog 1 channel selection**

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit
None	x	0

Table 78. Analog watchdog 1 channel selection (continued)

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit
All channels	0	1
Single <sup>(1)</sup> channel	1	1

1. Selected by the AWD1CH[4:0] bits

### 15.7.2 Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the AWDxCHy in ADC\_AWDxCR (x = 2, 3).

The corresponding watchdog is enabled when any AWDxCHy bit (x = 2,3) is set in ADC\_AWDxCR register.

When converting data with a resolution of less than 12 bits (configured through DRES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

[Table 77](#) 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.

### 15.7.3 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 52](#))

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  $f_{ADC}$  clock 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  $f_{ADC}$  clock cycles after the comparison.

As ADC\_AWDx\_OUT is generated by the  $f_{ADC}$  clock domain and AWD flag is generated by the APB clock domain, the rising edges of these signals are not synchronized.

Figure 50. ADC\_AWDx\_OUT signal generation

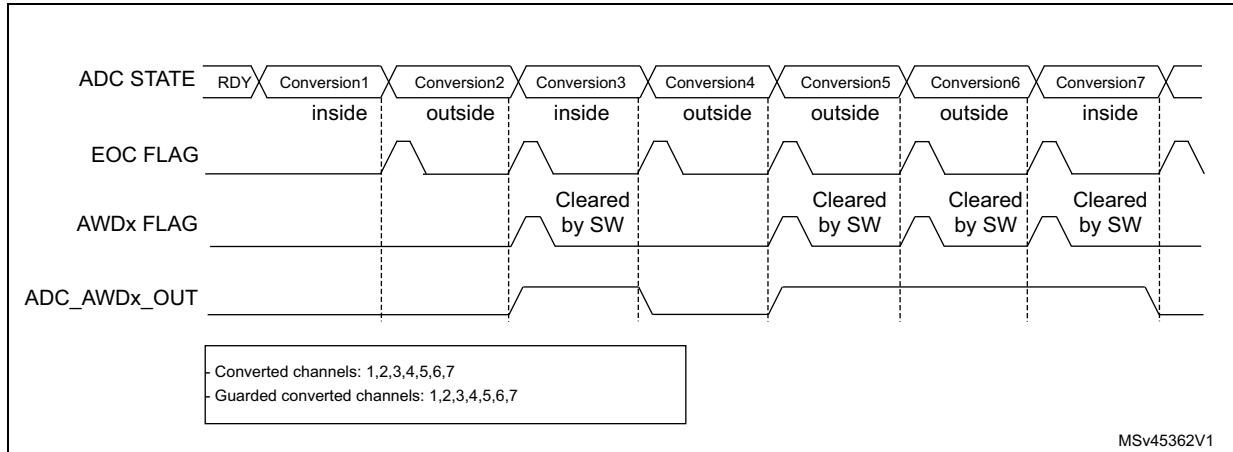


Figure 51. ADC\_AWDx\_OUT signal generation (AWDx flag not cleared by software)

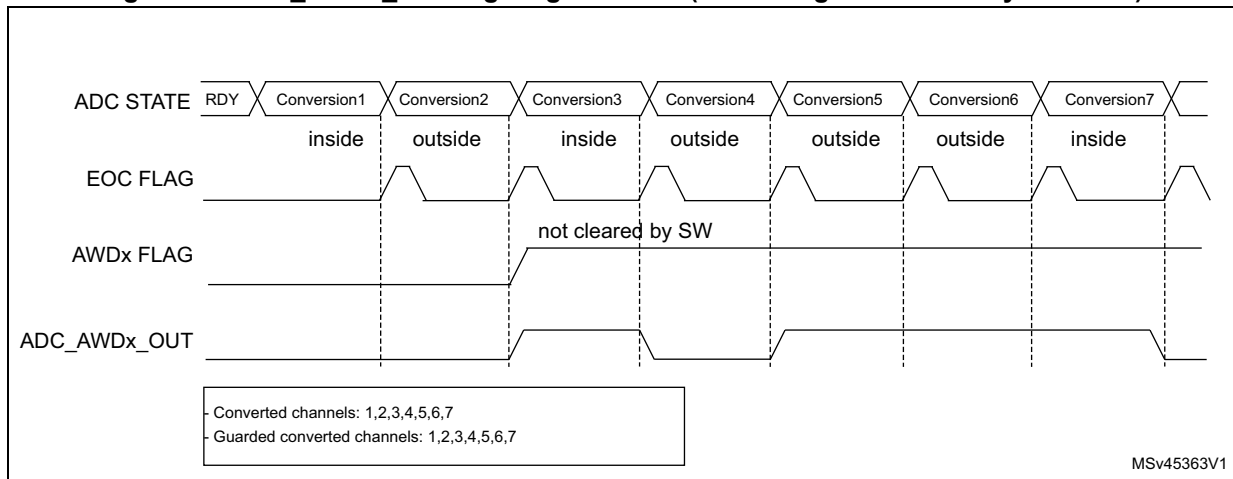
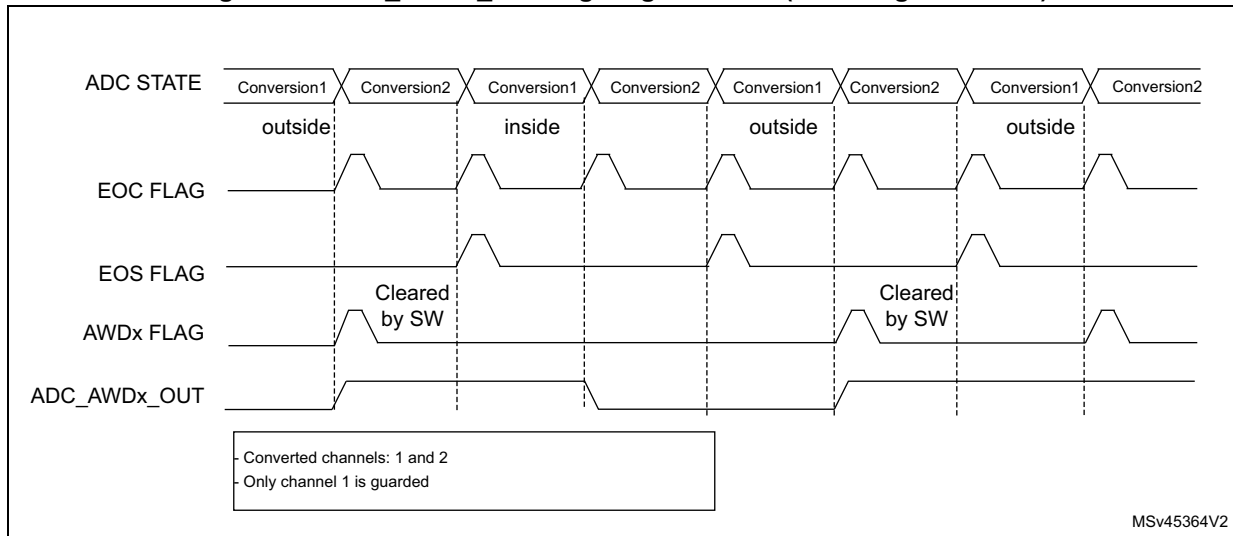


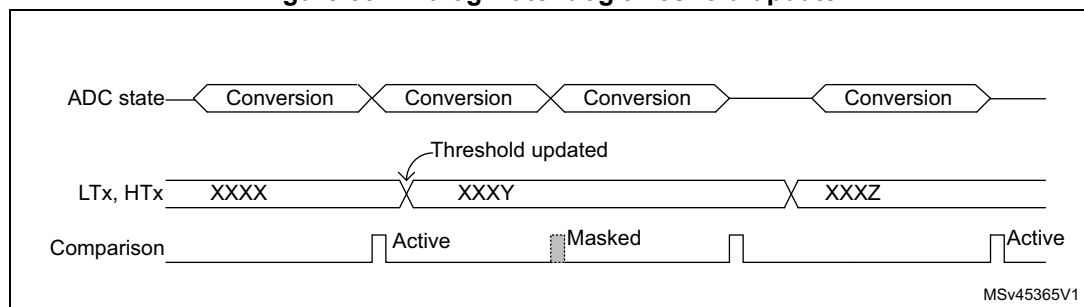
Figure 52. ADC\_AWDx\_OUT signal generation (on a single channel)



### 15.7.4 Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed during an analog-to-digital conversion (that is between the start of the conversion and the end of conversion of the ADC internal state). If HTx and LTx bits are programmed during the ADC guarded channel conversion, the watchdog function is masked for this conversion. This mask is cleared when starting a new conversion, and the resulting new AWD threshold 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 generated any interrupt or an ADC\_AWDx\_OUT signal. The Interrupt and the ADC\_AWDx\_OUT generation only occurs at the end of the ADC conversion that started after the threshold update. If ADC\_AWDx\_OUT is already asserted, programming the new threshold does not deassert the ADC\_AWDx\_OUT signal.

Figure 53. Analog watchdog threshold update



### 15.8 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

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 OVSR[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 54. 20-bit to 16-bit result truncation**

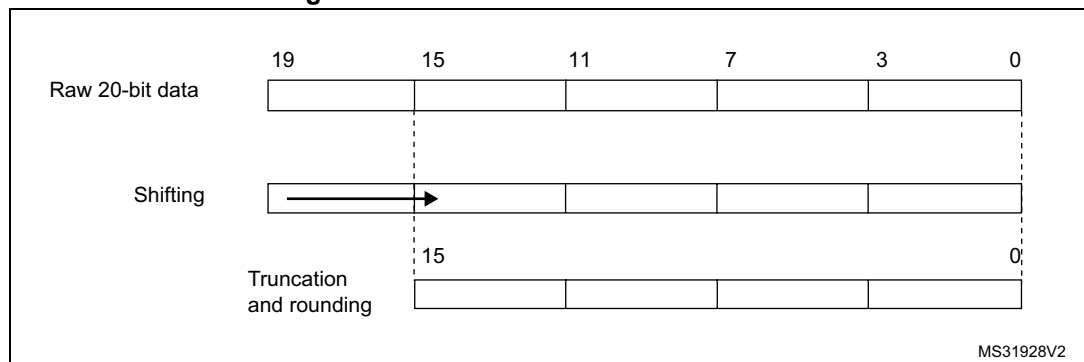


Figure 55 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 55. Numerical example with 5-bit shift and rounding**

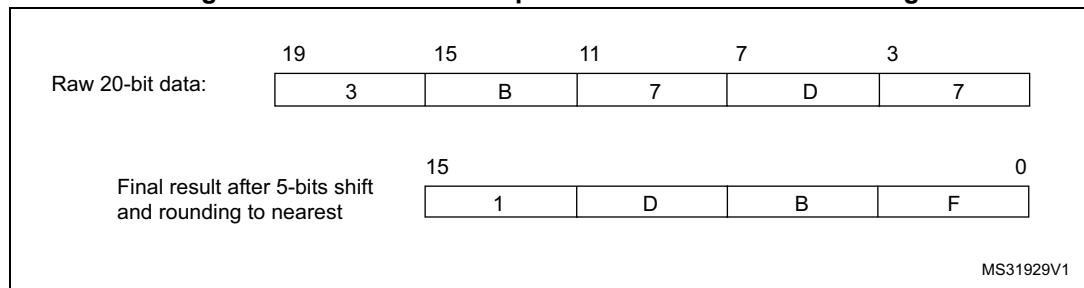




Table 79 gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFFF.

**Table 79. 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	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

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 x sequence length conversions total)

### 15.8.1 ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are available:

- Single or continuous mode conversions, forward or backward scanned sequences and up to 8 channels programmed sequence
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (WAIT, AUTOFF)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC\_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

*Note:* The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC\_CFGR1 is ignored and the data are always provided right-aligned.

### 15.8.2 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.*

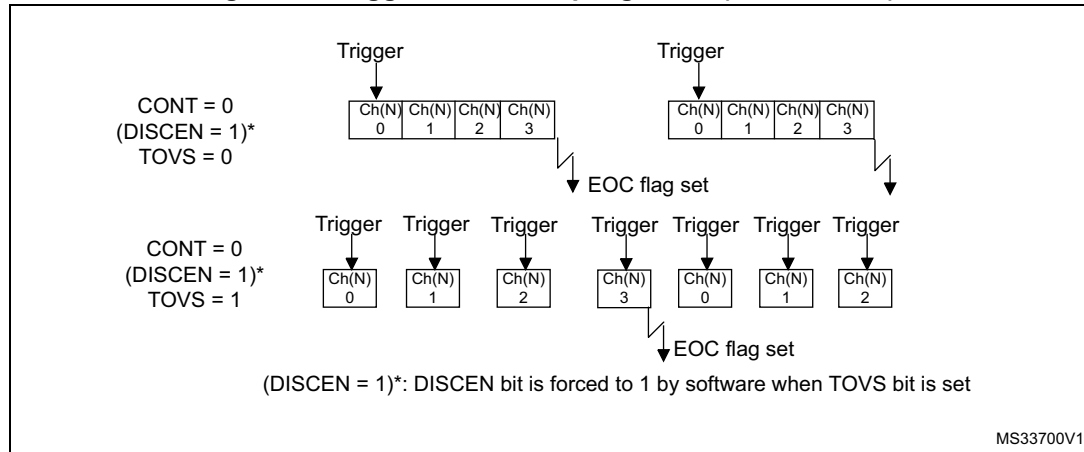
### 15.8.3 Triggered mode

The averager can also be used for basic filtering purposes. Although not a very efficient filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TOVS bit in ADC\_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 56 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 56. Triggered oversampling mode (TOVS bit = 1)



## 15.9 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the junction temperature ( $T_J$ ) of the device. The temperature sensor is internally connected to the ADC  $V_{IN}[12]$  input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor analog pin must be greater than the minimum  $T_{S\_temp}$  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 and comparators.  $V_{REFINT}$  is internally connected to the ADC  $V_{IN}[13]$  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 57 shows the block diagram of connections between the temperature sensor, the internal voltage reference and the ADC.

The TSEN bit must be set to enable the conversion of ADC  $V_{IN}[12]$  (temperature sensor) and the VREFEN bit must be set to enable the conversion of ADC  $V_{IN}[13]$  ( $V_{REFINT}$ ).

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.

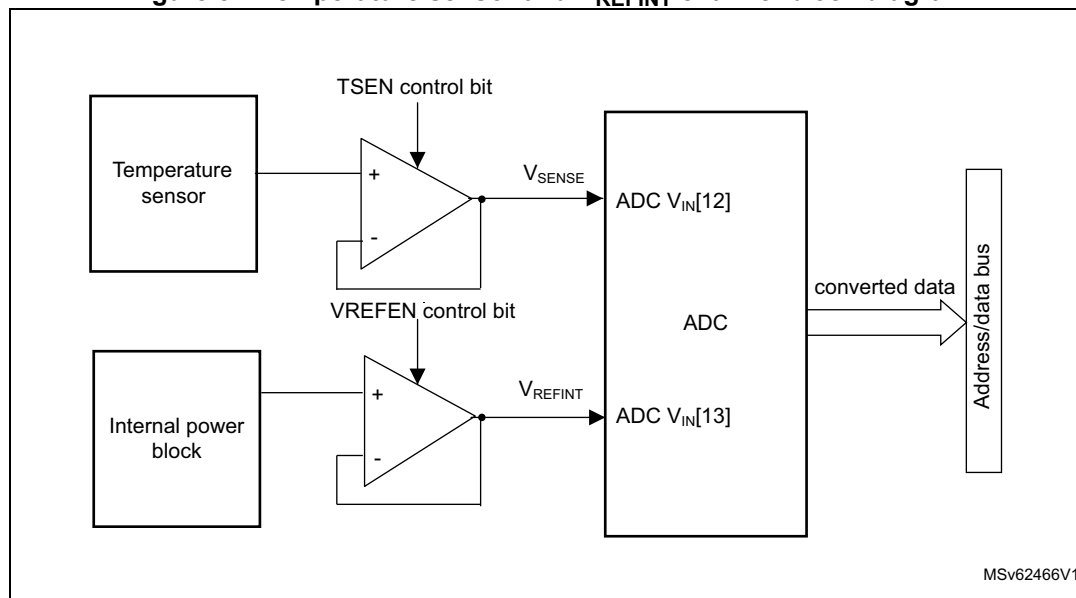
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference. Refer to the datasheet for additional information.

*Note:* Before entering any Stop mode, the temperature sensor and the internal reference voltage must be disabled by clearing TSEN and VREFEN, respectively.

**Main features**

- Linearity:  $\pm 2$  °C max, precision depending on calibration

**Figure 57. Temperature sensor and  $V_{REFINT}$  channel block diagram**



MSv62466V1

### Reading the temperature

1. Select the ADC  $V_{IN}[12]$  input channel.
2. Select an appropriate sampling time specified in the device datasheet ( $T_{S\_temp}$ ).
3. Set the TSEN bit in the ADC\_CCR register to wake up the temperature sensor from power down mode and wait for its stabilization time ( $t_{START}$ ).
4. Start the ADC conversion by setting the ADSTART bit in the ADC\_CR register (or by external trigger).
5. Read the resulting  $V_{SENSE}$  data in the ADC\_DR register.
6. Calculate the temperature using the following formula

$$\text{Temperature (in } ^\circ\text{C)} = \frac{TS\_CAL2\_TEMP - TS\_CAL1\_TEMP}{TS\_CAL2 - TS\_CAL1} \times (TS\_DATA - TS\_CAL1) + TS\_CAL1\_TEMP$$

Where:

- $TS\_CAL2$  is the temperature sensor calibration value acquired at  $TS\_CAL2\_TEMP$  (refer to the datasheet for  $TS\_CAL2$  value)
- $TS\_CAL1$  is the temperature sensor calibration value acquired at  $TS\_CAL1\_TEMP$  (refer to the datasheet for  $TS\_CAL1$  value)
- $TS\_DATA$  is the actual temperature sensor output value converted by ADC  
Refer to the specific device datasheet for more information about  $TS\_CAL1$  and  $TS\_CAL2$  calibration points.

*Note:* The sensor has a startup time after waking from power down mode before it can output  $V_{SENSE}$  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 $V_{REF+}$ voltage using the internal reference voltage

$V_{REF+}$  voltage may be subject to variation or not precisely known. The embedded internal reference voltage ( $V_{REFINT}$ ) and its calibration data acquired by the ADC during the manufacturing process at  $V_{REF+_charac}$  can be used to evaluate the actual  $V_{REF+}$  voltage level.

The following formula gives the actual  $V_{REF+}$  voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times V_{REFINT\_CAL} / V_{REFINT\_DATA}$$

Where:

- $V_{REF+_Charac}$  is the value of  $V_{REF+}$  voltage characterized at  $V_{REFINT}$  during the manufacturing process. It is specified in the device datasheet.
- $V_{REFINT\_CAL}$  is the  $V_{REFINT}$  calibration value
- $V_{REFINT\_DATA}$  is the actual  $V_{REFINT}$  output value converted by ADC

### Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the analog power supply and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of  $V_{REF+}$ . For applications where  $V_{REF+}$  is known and ADC converted values are right-aligned you can use the following formula to get this absolute value:

$$V_{\text{CHANNEL}x} = \frac{V_{\text{REF}+}}{\text{NUM\_CODES}} \times \text{ADC\_DATA}_x$$

For applications where  $V_{\text{REF}+}$  value is not known, you must use the internal voltage reference and  $V_{\text{REF}+}$  can be replaced by the expression provided in [Calculating the actual  \$V\_{\text{REF}+}\$  voltage using the internal reference voltage](#), resulting in the following formula:

$$V_{\text{CHANNEL}x} = \frac{V_{\text{REF}+ \text{ Charac}} \times \text{VREFINT\_CAL} \times \text{ADC\_DATA}_x}{\text{VREFINT\_DATA} \times \text{NUM\_CODES}}$$

Where:

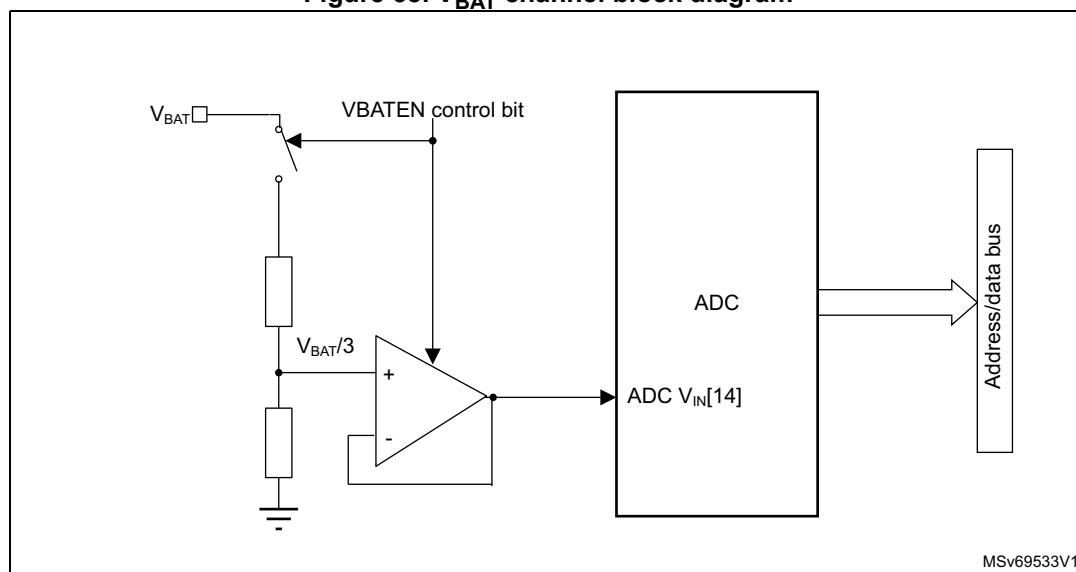
- $V_{\text{REF}+ \text{ Charac}}$  is the value of  $V_{\text{REF}+}$  voltage characterized at  $V_{\text{REFINT}}$  during the manufacturing process. It is specified in the device datasheet.
- $\text{VREFINT\_CAL}$  is the  $\text{VREFINT}$  calibration value
- $\text{ADC\_DATA}_x$  is the value measured by the ADC on channel  $x$  (right-aligned)
- $\text{VREFINT\_DATA}$  is the actual  $\text{VREFINT}$  output value converted by the ADC
- $\text{NUM\_CODES}$  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.

## 15.10 Battery voltage monitoring

The  $\text{VBATEN}$  bit in the  $\text{ADC\_CCR}$  register allows the application to measure the backup battery voltage on the  $\text{VBAT}$  pin. As the  $V_{\text{BAT}}$  voltage can be higher than  $V_{\text{REF}+}$ , to ensure the correct operation of the ADC, the  $\text{VBAT}$  pin is internally connected to a bridge divider. This bridge is automatically enabled when  $\text{VBATEN}$  is set, to connect  $V_{\text{BAT}}$  to the ADC  $V_{\text{IN}}[14]$  input channel. As a consequence, the converted digital value is  $V_{\text{BAT}}/3$ . To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed for ADC conversion.

Figure 58.  $V_{\text{BAT}}$  channel block diagram



## 15.11 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 80. 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

## 15.12 ADC registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

### 15.12.1 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.	CCRD Y	Res.	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSM P	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\_CHSELR 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\_CHSELR or changing CHSELRMOD or SCANDIR), it must wait until the CCRDY flag rises before configuring again or starting conversions, otherwise the new configuration (or the START bit) 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 it.

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

### 15.12.2 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	EOSM PIE	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).*

### 15.12.3 ADC control register (ADC\_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	Res.	ADVRE GEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	ADSTA RT	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  $t_{\text{ADCVREG\_STUP}}$ .

It is cleared by software to disable the voltage regulator. It can be cleared only if ADEN is cleared.

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 EXTEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– In single conversion mode (CONT = 0, DISCEN = 0), when software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion Sequence (EOS) flag.

– In discontinuous conversion mode (CONT = 0, DISCEN = 1), when the software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion (EOC) flag.

– In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

0: No ADC conversion is ongoing.

1: Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.

*Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).*

*After writing to ADC\_CHSELR register or changing 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 ADCAL = 0, ADSTP = 0, ADSTART = 0, ADDIS = 0, ADEN = 0, and ADVREGEN = 1.*

**15.12.4 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.	AWD1CH[4:0]					Res.	Res.	AWD1EN	AWD1SGL	CHSELRMOD	Res.	Res.	Res.	Res.	DISCEN
	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
AUTOFF	WAIT	CONT	OVROD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	RES[1:0]		SCANDIR	DMACFG	DMAEN
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 30:26 **AWD1CH[4:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input Channel 0 monitored by AWD

00001: ADC analog input Channel 1 monitored by AWD

.....

10001: ADC analog input Channel 17 monitored by AWD

10010: ADC analog input Channel 18 monitored by AWD

Others: Reserved

*Note: The channel selected by the AWDCH[4:0] bits must be also set into the CHSELR register.*

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

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

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

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

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.

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

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 73: External triggers](#) for details):

000: TRG0

001: TRG1

010: TRG2

011: TRG3

100: TRG4

101: TRG5

110: TRG6

111: TRG7

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to [Figure 44: Data alignment and resolution \(oversampling disabled: OVSE = 0\)](#) on page 362

0: Right alignment

1: Left alignment

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 is scanned in the sequence. It is effective only if CHSELMOD bit is cleared.

0: Upward scan (from CHSEL0 to CHSEL18)

1: Backward scan (from CHSEL18 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 15.5.5: Managing converted data using the DMA on page 364](#).

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 15.5.5: Managing converted data using the DMA on page 364](#).

- 0: DMA disabled
- 1: DMA enabled

### 15.12.5 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.
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 by 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.*

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 field defines the number of oversampling ratio.

000: 2x

001: 4x

010: 8x

011: 16x

100: 32x

101: 64x

110: 128x

111: 256x

Starting from 32x, it is mandatory to use OVSS[3:0] to reduce the data size to less than 16 bits to fit to the ADC\_DR register. For example, if OVSR[2:0] is configured to 32x, the data width is 17 bits, and OVSS[3:0] must be set to 1-bit shift.

*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.*



### 15.12.6 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.	Res.	Res.	Res.	Res.	SMPSEL18	SMPSEL17	SMPSEL16	SMPSEL15	SMPSEL14	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8
					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
SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2[2:0]			Res.	SMP1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

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

Bits 26:8 **SMPSELx**: Channel-x sampling time selection (x = 18 to 0)

These bits are written by software to define which sampling time is used.

0: Sampling time of CHANNELx use the setting of SMP1[2:0] register.

1: Sampling time of CHANNELx use the setting of SMP2[2:0] register.

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMP2[2:0]**: Sampling time selection 2

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 160.5 ADC clock cycles

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMP1[2:0]**: Sampling time selection 1

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 160.5 ADC clock cycles

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

### 15.12.7 ADC watchdog threshold register (ADC\_AWD1TR)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:28 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 15.7: Analog window watchdogs on page 367](#).

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 15.7: Analog window watchdogs on page 367](#).

### 15.12.8 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]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT2[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:28 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 15.7: Analog window watchdogs on page 367](#).

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 15.7: Analog window watchdogs on page 367](#).

### 15.12.9 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.	Res.	Res.	Res.	Res.	CHSEL 18	CHSEL 17	CHSEL 16
													rW	rW	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 31:19 Reserved, must be kept at reset value.

Bits 18:0 **CHSEL[18:0]**: 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 36: 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.*

**15.12.10 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	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	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).*

### 15.12.11 ADC watchdog threshold register (ADC\_AWD3TR)

Address offset: 0x2C

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT3[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.	LT3[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 **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 15.7: Analog window watchdogs on page 367](#).

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

Bits 11:0 **LT3[11:0]**: Analog watchdog 3lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to [Section 15.7: Analog window watchdogs on page 367](#).

### 15.12.12 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 44](#).

Just after a calibration is complete, DATA[6:0] contains the calibration factor.

### 15.12.13 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.	Res.	Res.	Res.	Res.	AWD2 CH18	AWD2 CH17	AWD2 CH16
													r/w	r/w	r/w
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
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 18:0 **AWD2CH[18:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 2 (AWD2).

0: ADC analog channel-x is not monitored by AWD2

1: ADC analog channel-x is monitored by AWD2

*Note:* The channels selected through ADC\_AWD2CR must be also configured into the ADC\_CHSELR registers. The software is allowed to write this bit only when ADEN = 0.

### 15.12.14 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.	Res.	Res.	Res.	Res.	AWD3 CH18	AWD3 CH17	AWD3 CH16
													rw	rw	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 31:19 Reserved, must be kept at reset value.

Bits 18:0 **AWD3CH[18:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 3 (AWD3).

0: ADC analog channel-x is not monitored by AWD3

1: ADC analog channel-x is monitored by AWD3

*Note: The channels selected through ADC\_AWD3CR must be also configured into the ADC\_CHSELR registers. The software is allowed to write this bit only when ADEN = 0.*

### 15.12.15 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.	CALFACT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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).*

**15.12.16 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.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	PRESC[3:0]				Res.	Res.
							rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

Bit 24 **VBATEN**: V<sub>BAT</sub> enable

This bit is set and cleared by software to enable/disable the V<sub>BAT</sub> channel.

0: V<sub>BAT</sub> channel disabled, DAC\_OUT2 connected to ADC channel 14

1: V<sub>BAT</sub> channel enabled

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)*

Bit 23 **TSEN**: Temperature sensor buffer enable

This bit is set and cleared by software to enable/disable the temperature sensor buffer.

0: Temperature sensor buffer disabled, DAC\_OUT1 connected to ADC channel 12

1: Temperature sensor buffer enabled

*Note: Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

*This bit must be cleared before entering low-power modes to avoid unwanted power consumption.*

Bit 22 **VREFEN**: V<sub>REFINT</sub> buffer enable

This bit is set and cleared by software to enable/disable the V<sub>REFINT</sub> buffer.

0: V<sub>REFINT</sub> buffer disabled

1: V<sub>REFINT</sub> buffer enabled

*Note: Software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

*This bit must be cleared before entering low-power modes to avoid unwanted power consumption.*



Bits 21:18 **PRESC[3:0]**: ADC prescaler

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

- 0000: input ADC clock not divided
- 0001: input ADC clock divided by 2
- 0010: input ADC clock divided by 4
- 0011: input ADC clock divided by 6
- 0100: input ADC clock divided by 8
- 0101: input ADC clock divided by 10
- 0110: input ADC clock divided by 12
- 0111: input ADC clock divided by 16
- 1000: input ADC clock divided by 32
- 1001: input ADC clock divided by 64
- 1010: input ADC clock divided by 128
- 1011: input ADC clock divided by 256
- Other: Reserved

*Note: Software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).*

Bits 17:0 Reserved, must be kept at reset value.

### 15.13 ADC register map

The following table summarizes the ADC registers.

**Table 81. ADC 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>ADC_ISR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRDY	Res.	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	<b>ADC_IER</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRDYIE	Res.	EOCALIE	Res.	AWD3IE	AWD2IE	AWD1IE	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	<b>ADC_CR</b>	ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
	Reset value	0			0																								0		0	0	0	0
0x0C	<b>ADC_CFGR1</b>	Res.	AWDCH[4:0]				Res.	Res.	Res.	Res.	AWD1EN	AWD1SGL	CHSELRMOD	Res.	Res.	Res.	Res.	DISCEN	AUTOFF	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL [2:0]		ALIGN	RES [1:0]	Res.	SCANDIR	DMACFG	DMAEN	
	Reset value		0	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 81. ADC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
0x10	ADC_CFGR2	CKMODE[1:0]		LFTRIG											Res.											TOVS			OVSS[3:0]			OVSR[2:0]			Res.	OVSE
	Reset value	0	0	0																					0	0	0	0	0	0	0	0	0	0	0	
0x14	ADC_SMPR	Res.	Res.	Res.	Res.	Res.	SMPSEL18	SMPSEL17	SMPSEL16	SMPSEL15	SMPSEL14	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8	SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2 [2:0]			Res.	SMP1 [2:0]			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	
0x18	Reserved	Reserved																																		
0x1C	Reserved	Reserved																																		
0x20	ADC_AWD1TR	Res.	Res.	Res.	Res.	HT1[11:0]											Res.	Res.	Res.	Res.	LT1[11:0]															
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0	0	0	
0x24	ADC_AWD2TR	Res.	Res.	Res.	Res.	HT2[11:0]											Res.	Res.	Res.	Res.	LT2[11:0]															
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0	0	0	
0x28	ADC_CHSELR (CHSELRMOD=0)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL18	CHSEL17	CHSEL16	CHSEL15	CHSEL14	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSELO		
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x28	ADC_CHSELR (CHSELRMOD=1)	SQ8[3:0]			SQ7[3:0]			SQ6[3:0]			SQ5[3:0]			SQ4[3:0]			SQ3[3:0]			SQ2[3:0]			SQ1[3:0]													
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x2C	ADC_AWD3TR	Res.	Res.	Res.	Res.	HT3[11:0]											Res.	Res.	Res.	Res.	LT3[11:0]															
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0	0	0	
0x30 0x34 0x38 0x3C	Reserved	Reserved																																		
0x40	ADC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]																		
	Reset value																																			
...	Reserved	Reserved																																		
0xA0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 81. ADC register map and reset values (continued)**

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0xA4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	
...	Reserved	Reserved																																
0xB4	ADC_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
...	Reserved	Reserved																																
0x308	ADC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																		

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

## 16 Digital-to-analog converter (DAC)

### 16.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DACx\_OUTy pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the sample and hold mode.

### 16.2 DAC main features

The DAC main features are the following (see [Figure 59: Dual-channel DAC block diagram](#))

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx\_OUTy output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREF+ pin or internal VREFBUF reference

[Figure 59](#) shows the block diagram of a DAC channel and [Table 83](#) gives the pin description.

## 16.3 DAC implementation

Table 82. DAC features

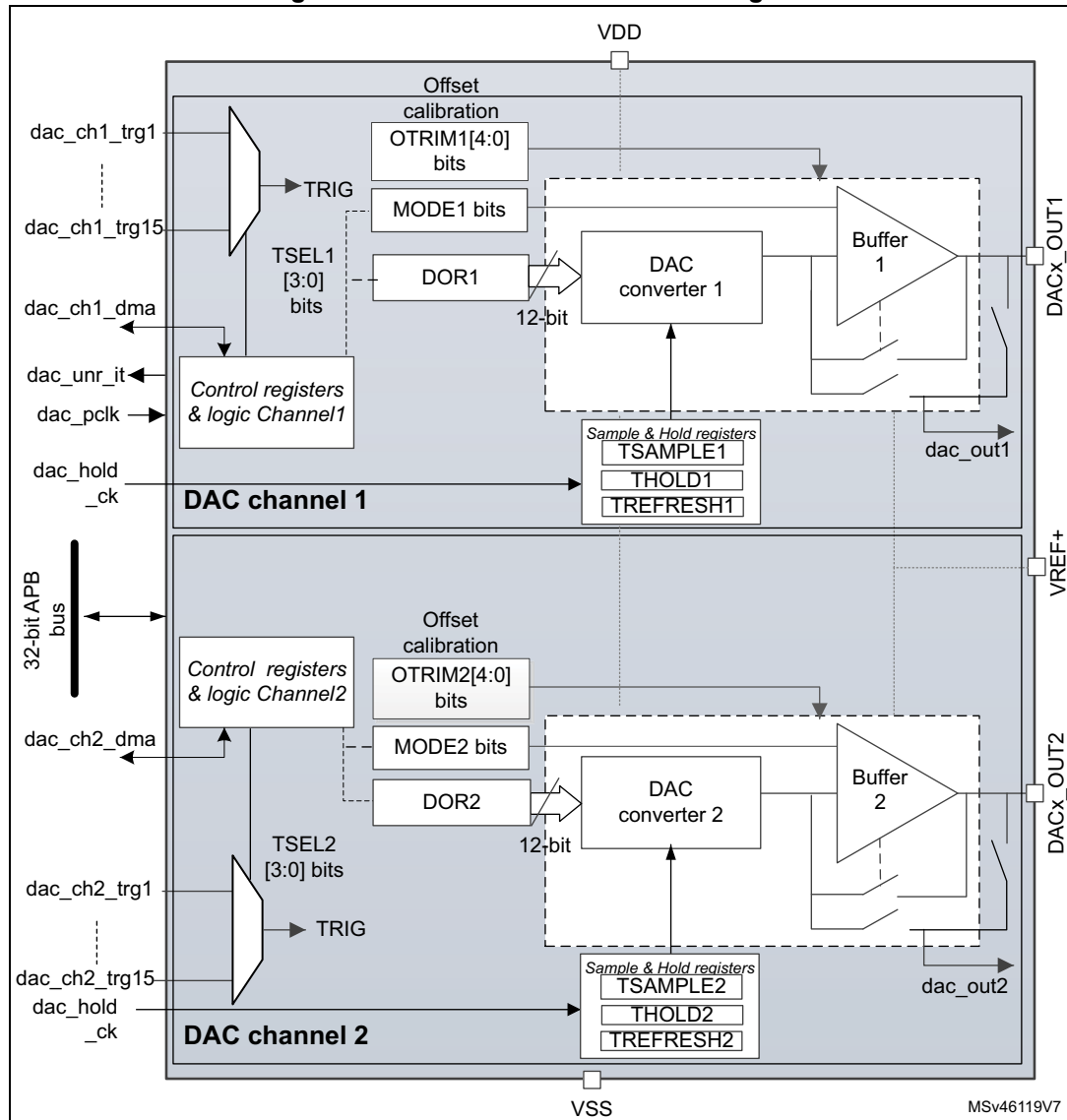
DAC features	DAC1 <sup>(1)</sup>
Dual channel	X
Output buffer	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5
Maximum sampling time	1 Msps
Autonomous mode	-
VREF+ pin	X

1. There is no DAC on STM32G031xx and STM32G041xx.

## 16.4 DAC functional description

### 16.4.1 DAC block diagram

Figure 59. Dual-channel DAC block diagram



1. `MODEx` bits in the `DAC_MCR` control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.
2. Refer to [Section 16.3: DAC implementation](#) for channel2 availability.

## 16.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx\_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac\_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI clock source (dac\_hold\_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx\_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI clock source (dac\_hold\_ck).

**Table 83. DAC input/output pins**

Pin name	Signal type	Remarks
VREF+	Input, analog positive reference	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDD	Input, analog supply	Analog power supply
VSS	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

**Table 84. DAC internal input/output signals**

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_pclk	Input	DAC peripheral clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 85. DAC interconnection

Signal name	Source	Type
dac_hold_ck	ck_lsi (selected in the RCC)	LSI clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	tim1_trgo	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	tim2_trgo	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	tim3_trgo	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	tim6_trgo	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	tim7_trgo	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	tim15_trgo	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	lptim1_out	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	lptim2_out	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	exti9	External pin

### 16.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC\_CR register. The DAC channel is then enabled after a  $t_{WAKEUP}$  startup time.

*Note:* The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

### 16.4.4 DAC data format

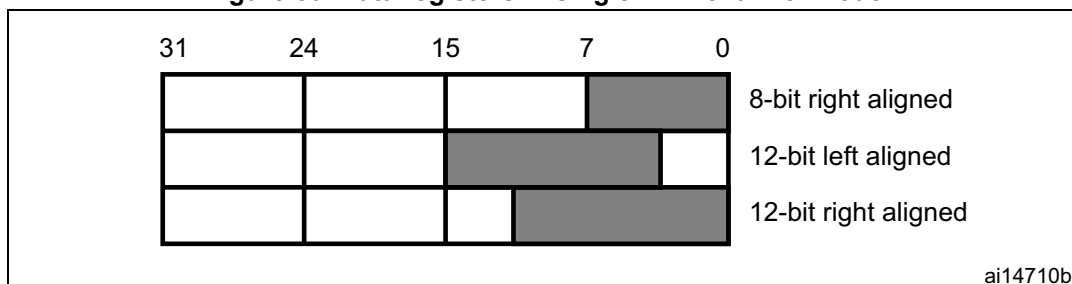
Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel
  - There are three possibilities:
    - 8-bit right alignment: the software has to load data into the DAC\_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)
    - 12-bit left alignment: the software has to load data into the DAC\_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
    - 12-bit right alignment: the software has to load data into the DAC\_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC\_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC\_DHRx (data holding registerx, which are internal non-memory-mapped registers). The DAC\_DHRx register is then loaded into the DAC\_DORx register either automatically, by software trigger or by an external event trigger.



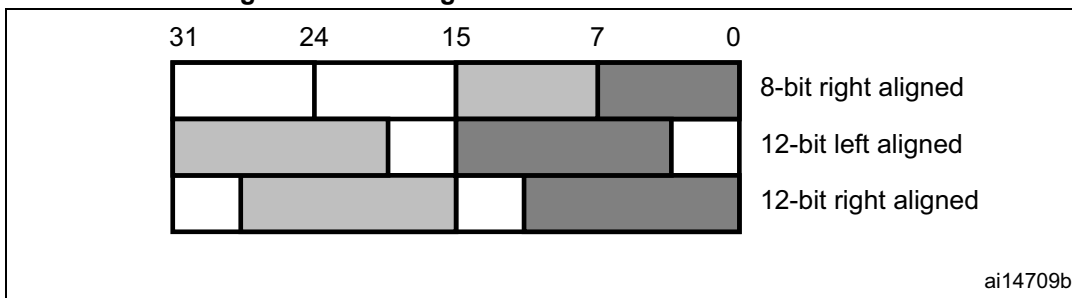
Figure 60. Data registers in single DAC channel mode



- Dual DAC channels (when available)  
 There are three possibilities:
  - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC\_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC\_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
  - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC\_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC\_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC\_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC\_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC\_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC\_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 61. Data registers in dual DAC channel mode



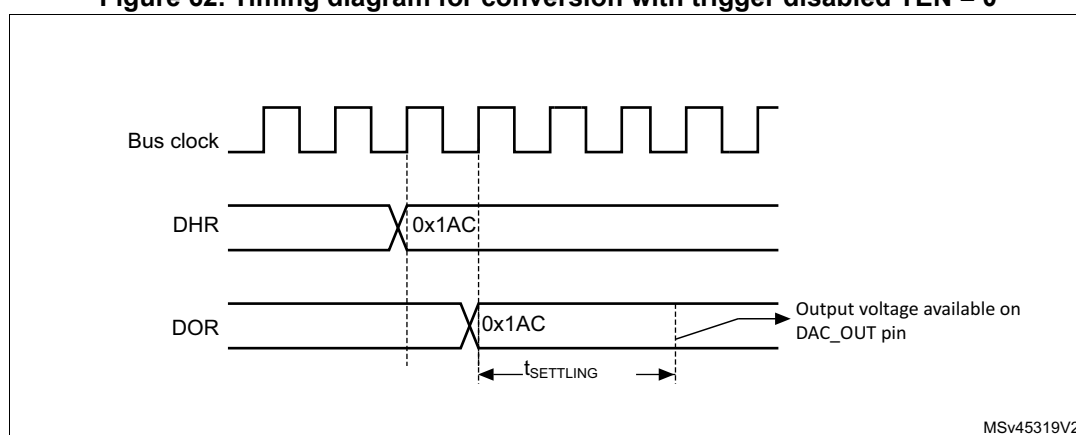
### 16.4.5 DAC conversion

The DAC\_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC\_DHRx register (write operation to DAC\_DHR8Rx, DAC\_DHR12Lx, DAC\_DHR12Rx, DAC\_DHR8RD, DAC\_DHR12RD or DAC\_DHR12LD).

Data stored in the DAC\_DHRx register are automatically transferred to the DAC\_DORx register after one dac\_pclk clock cycle, if no hardware trigger is selected (TENx bit in DAC\_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC\_CR register is set) and a trigger occurs, the transfer is performed three dac\_pclk clock cycles after the trigger signal.

When DAC\_DORx is loaded with the DAC\_DHRx contents, the analog output voltage becomes available after a time  $t_{\text{SETTLING}}$  that depends on the power supply voltage and the analog output load.

**Figure 62. Timing diagram for conversion with trigger disabled TEN = 0**



### 16.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and  $V_{\text{REF+}}$ .

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DAC output} = V_{\text{REF}} \times \frac{\text{DOR}}{4096}$$

where all voltages are expressed in Volt.

### 16.4.7 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC\_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in [Section 16.4.2](#).

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC\_DHRx register are transferred into the DAC\_DORx register. The DAC\_DORx register is updated three dac\_pclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC\_DORx register has been loaded with the DAC\_DHRx register contents.

*Note:* TSELx[3:0] bit cannot be changed when the ENx bit is set.

*When software trigger is selected, the transfer from the DAC\_DHRx register to the DAC\_DORx register takes only one dac\_pclk clock cycle.*

### 16.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC\_DHRx register is transferred into the DAC\_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC\_DHRx to DAC\_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC\_DHRx before the first trigger event occurs.

#### DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC\_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

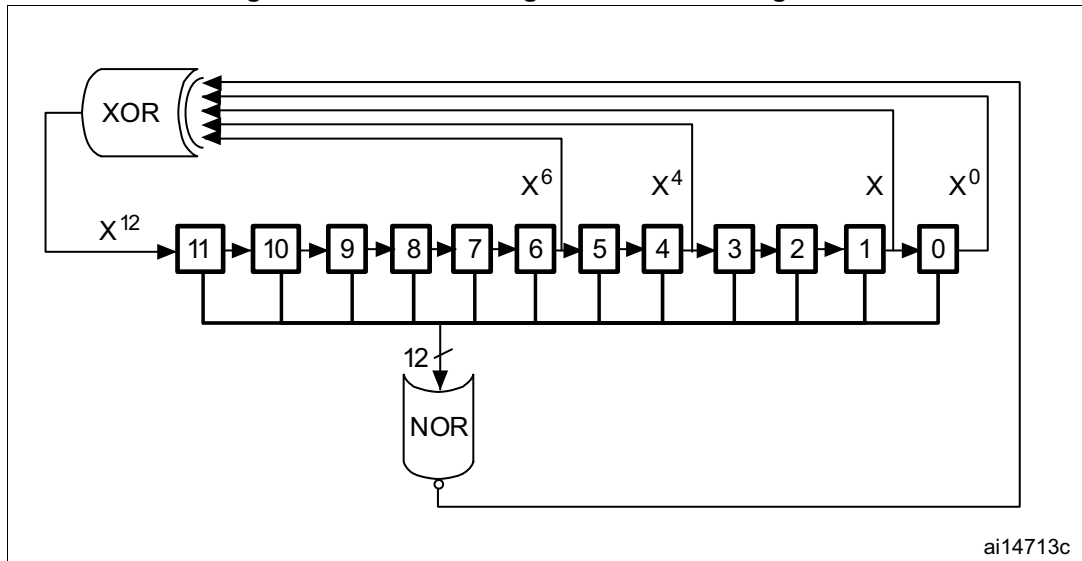
The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC\_CR register is enabled.

### 16.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac\_pclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 63. DAC LFSR register calculation algorithm

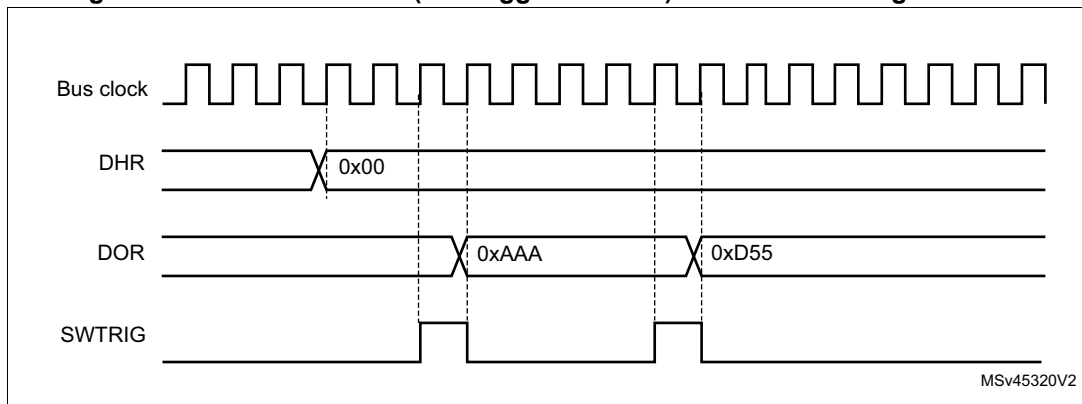


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC\_CR register, is added up to the DAC\_DHRx contents without overflow and this value is then transferred into the DAC\_DORx register.

If LFSR is 0x0000, a 1 is injected into it (antiloop-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 64. DAC conversion (SW trigger enabled) with LFSR wave generation



**Note:** The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC\_CR register.

### 16.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC\_CR register. An internal triangle counter is incremented three dac\_pclk clock cycles after each trigger event. The value of this counter is then added to the DAC\_DHRx register without overflow and the sum is transferred into the DAC\_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 65. DAC triangle wave generation

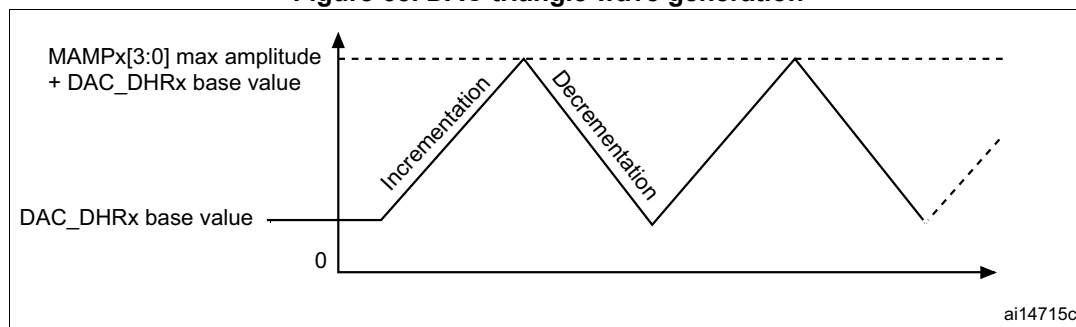
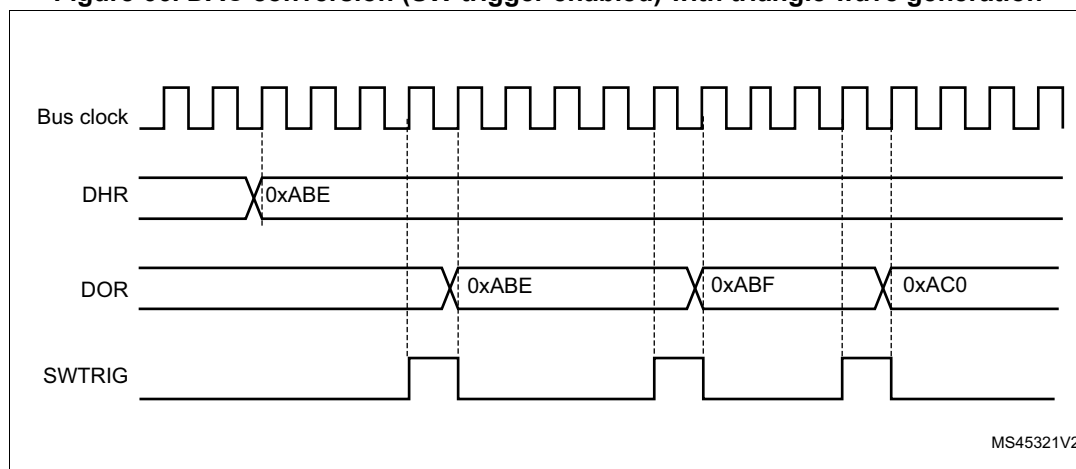


Figure 66. DAC conversion (SW trigger enabled) with triangle wave generation



**Note:** The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC\_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

### 16.4.11 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

#### Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx\_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC\_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC\_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

#### Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI low-speed clock (dac\_hold\_ck) in addition to the dac\_pclk clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI low-speed clock (dac\_hold\_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC\_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC\_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC\_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC\_SHRR register

The timings for the three phases above are in units of LSI clock periods. As an example, to configure a sample time of 350  $\mu\text{s}$ , a hold time of 2 ms and a refresh time of 100  $\mu\text{s}$  assuming LSI ~32 KHz is selected:

- 12 cycles are required for sample phase:  $T_{\text{SAMPLEx}[9:0]} = 11$ .
- 62 cycles are required for hold phase:  $T_{\text{HOLDx}[9:0]} = 62$ .
- and 4 cycles are required for refresh period:  $T_{\text{REFRESHx}[7:0]} = 4$ .

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

**Table 86. Sample and refresh timings**

Buffer State	$t_{\text{SAMP}}^{(1)(2)}$	$t_{\text{REFRESH}}^{(2)(3)}$
Enable	$7 \mu\text{s} + (10 \cdot R_{\text{BON}} \cdot C_{\text{SH}})$	$7 \mu\text{s} + (R_{\text{BON}} \cdot C_{\text{SH}}) \cdot \ln(2 \cdot N_{\text{LSB}})$
Disable	$3 \mu\text{s} + (10 \cdot R_{\text{BOFF}} \cdot C_{\text{SH}})$	$3 \mu\text{s} + (R_{\text{BOFF}} \cdot C_{\text{SH}}) \cdot \ln(2 \cdot N_{\text{LSB}})$

1. In the above formula, the settling to the desired code value with  $\frac{1}{2}$  LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2.  $C_{\text{SH}}$  is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase "Vd" is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with  $\frac{1}{2}$  LSB error accuracy requires  $\ln(2 \cdot N_{\text{lsb}})$  constant time of the DAC.

### Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

$$C_{\text{SH}} = 100 \text{ nF}$$

$$V_{\text{DD}} = 3.0 \text{ V}$$

Sampling phase:

$$t_{\text{SAMP}} = 7 \mu\text{s} + (10 \cdot 2000 \cdot 100 \cdot 10^{-9}) = 2.007 \text{ ms}$$

(where  $R_{\text{BON}} = 2 \text{ k}\Omega$ )

Refresh phase:

$$t_{\text{REFRESH}} = 7 \mu\text{s} + (2000 \cdot 100 \cdot 10^{-9}) \cdot \ln(2 \cdot 10) = 606.1 \mu\text{s}$$

(where  $N_{\text{LSB}} = 10$  (10 LSB drop during the hold phase))

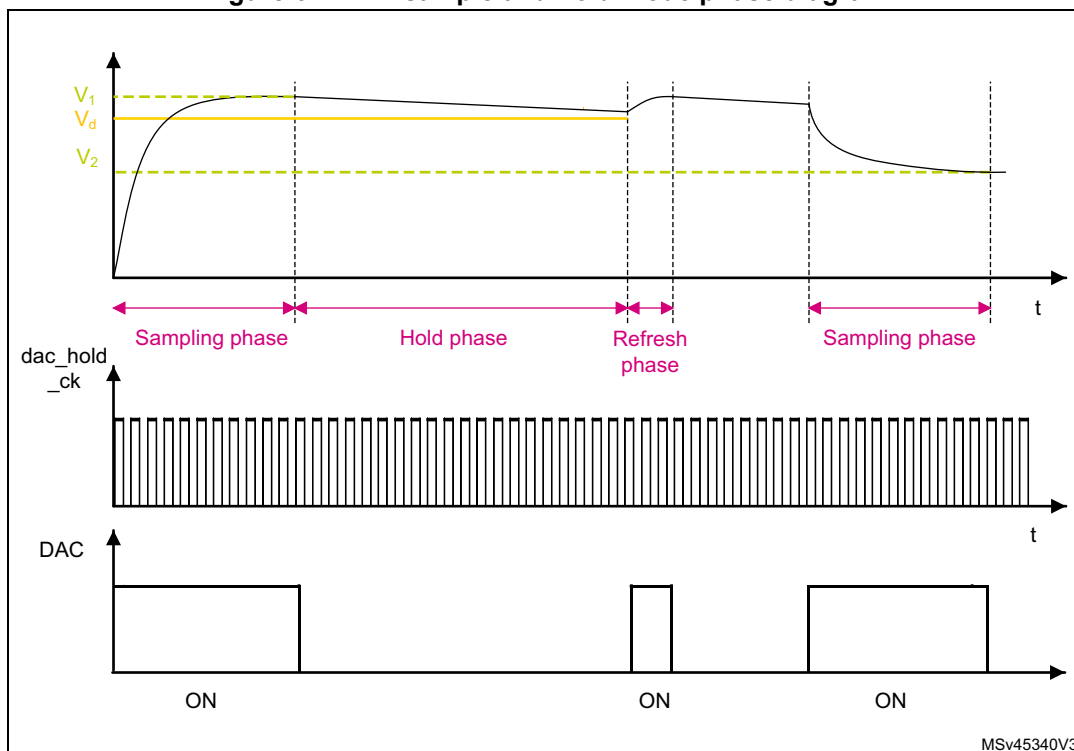
Hold phase:

$$D_v = i_{\text{leak}} \cdot t_{\text{hold}} / C_{\text{SH}} = 0.0073 \text{ V (10 LSB of 12bit at 3 V)}$$

$$i_{\text{leak}} = 150 \text{ nA (worst case on the IO leakage on all the temperature range)}$$

$$t_{\text{hold}} = 0.0073 \cdot 100 \cdot 10^{-9} / (150 \cdot 10^{-9}) = 4.867 \text{ ms}$$

Figure 67. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC\_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC\_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals

When MODEx[2:0] bits are equal to 111, an internal capacitor,  $C_{Lint}$ , holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC\_DHRx immediately triggers a new sample phase.

Table 87. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)



**Table 87. Channel output modes summary (continued)**

MODEx[2:0]			Mode	Buffer	Output connections
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

**16.4.12 DAC channel buffer calibration**

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = \left( (D / 2^N) \times G \times V_{REF} \right) + V_{OS}$$

Where  $V_{OUT}$  is the analog output, D is the digital input, G is the gain,  $V_{REF}$  is the nominal full-scale voltage, and  $V_{OS}$  is the offset voltage. For an ideal DAC channel,  $G = 1$  and  $V_{OS} = 0$ .

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the  $V_{OS}$ , a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to  $V_{REF+}/2$  signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL\_FLAGx bit).

Two calibration techniques are provided:

- **Factory trimming (default setting)**  
The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC\_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- **User trimming**  
The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when  $V_{DDA}$  voltage, temperature,  $V_{REF+}$  values change and can be done at any point during application by software.

*Note:* Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when  $V_{DD}$  is removed (example the device enters in Standby or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC\_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC\_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC\_CR register to 1.
4. Apply a trimming algorithm:
  - a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
  - b) Wait for  $t_{TRIM}$  delay.
  - c) Check if CAL\_FLAGx bit in DAC\_SR is set to 1.
  - d) Until the CAL\_FLAGx is read as 1 or the maximum trimming code is reached, increment OTRIMx[4:0] and repeat substeps from (b) to (d).

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

*Note:* A  $t_{TRIM}$  delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL\_FLAGx bit in DAC\_SR register in order to get a correct value. This parameter is specified into datasheet electrical characteristics section.

*If  $V_{DDA}$ ,  $V_{REF+}$  and temperature conditions do not change during device operation while it enters more often in Standby and VBAT modes, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.*

*When CENx bit is set, it is not allowed to set ENx bit.*

### 16.4.13 DAC channel conversion modes

Four conversion modes are possible.

#### Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC\_DHR12R1, DAC\_DHR12L1 or DAC\_DHR8R1).

When a DAC channel trigger arrives, the DHRx register is transferred into DAC\_DORx (three dac\_pclk clock cycles later).

#### Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC\_DHR12R1, DAC\_DHR12L1 or DAC\_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three `dac_pclk` clock cycles later). Then the LFSR1 counter is updated.

#### Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, `TENx`.
2. Configure the trigger sources by setting different values in the `TSELx[3:0]` bits.
3. Configure the DAC channel `WAVEx[1:0]` bits as 1x and the same maximum amplitude value in the `MAMPx[3:0]` bits.
4. Load the DAC channel data into the desired DHR register (`DAC_DHR12R1`, `DAC_DHR12L1` or `DAC_DHR8R1`).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the `DHRx` register and the sum is transferred into `DAC_DOR1` (three `dac_pclk` clock cycles later). The DAC channel triangle counter is then updated.

#### Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in `STRSTTRIGSELx[3:0]` and `STINCTRIGSELx[3:0]` bits.
2. Configure the DAC channel `WAVEx[1:0]` bits to 11 and set the same `STRSTDATAx[11:0]`, `STINCDATAx[15:0]` and `STDIRx` values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the `DHRx` register and transfers it into `DAC_DOR1` (three APB clock cycles later).

### 16.4.14 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: `DHR8RD`, `DHR12RD` and `DHR12LD`. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to `DHRxxxD` registers are required. As a result, two output channels can be used either independently or simultaneously.

11 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate `DAC_DHRx` registers if needed.

All modes are described in the paragraphs below.

#### Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits `TEN1` and `TEN2`.
2. Configure different trigger sources by setting different values in the `TSEL1` and `TSEL2` bitfields.
3. Load the dual DAC channel data into the desired DHR register (`DAC_DHR12RD`, `DAC_DHR12LD` or `DAC_DHR8RD`).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC\_DOR1 (three `dac_pclk` clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC\_DOR2 (three `dac_pclk` clock cycles later).

### Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three `dac_pclk` clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three `dac_pclk` clock cycles later). Then the LFSR2 counter is updated.

### Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three `dac_pclk` clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three `dac_pclk` clock cycles later). Then the LFSR2 counter is updated.

### Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three dac\_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three dac\_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

### Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three dac\_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three dac\_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

### Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

In this configuration, one dac\_pclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC\_DOR1 and DAC\_DOR2, respectively.

### Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC\_DOR1 and DAC\_DOR2, respectively (after three `dac_pclk` clock cycles).

#### **Simultaneous trigger with single LFSR generation**

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three `dac_pclk` clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three `dac_pclk` clock cycles later). The LFSR2 counter is then updated.

#### **Simultaneous trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three `dac_pclk` clock cycles later). The LFSR1 counter is then updated.

At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three `dac_pclk` clock cycles later). The LFSR2 counter is then updated.

#### **Simultaneous trigger with single triangle generation**

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three `dac_pclk` clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three `dac_pclk` clock cycles later). The DAC channel2 triangle counter is then updated.

### Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three `dac_pclk` clock cycles later). Then the DAC channel2 triangle counter is updated.

## 16.5 DAC in low-power modes

**Table 88. Effect of low-power modes on DAC**

Mode	Description
Sleep	No effect, DAC used with DMA
Stop 0	The DAC remains active with a static value, if sample and hold mode is selected using LSI clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

## 16.6 DAC interrupts

Table 89. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

## 16.7 DAC registers

Refer to [Section 1 on page 55](#) for a list of abbreviations used in register descriptions.

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

### 16.7.1 DAC control register (DAC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC\_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in normal operating mode

1: DAC channel2 in calibration mode

*Note: This bit is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

*Note: This bit is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*



Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.

0: DAC channel2 DMA mode disabled

1: DAC channel2 DMA mode enabled

*Note: This bit is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

*Note: These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.

00: wave generation disabled

01: Noise wave generation enabled

1x: Triangle wave generation enabled

*Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)*

*These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2

0000: SWTRIG2

0001: dac\_ch2\_trg1

0010: dac\_ch2\_trg2

...

1111: dac\_ch2\_trg15

Refer to the trigger selection tables in [Section 16.4.2: DAC pins and internal signals](#) for details on trigger configuration and mapping.

*Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).*

*These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*

- Bit 17 **TEN2**: DAC channel2 trigger enable  
This bit is set and cleared by software to enable/disable DAC channel2 trigger  
0: DAC channel2 trigger disabled and data written into the DAC\_DHR2 register are transferred one `dac_pclk` clock cycle later to the DAC\_DOR2 register  
1: DAC channel2 trigger enabled and data from the DAC\_DHR2 register are transferred three `dac_pclk` clock cycles later to the DAC\_DOR2 register  
*Note:* When software trigger is selected, the transfer from the DAC\_DHR2 register to the DAC\_DOR2 register takes only one `dac_pclk` clock cycle.  
*These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*
- Bit 16 **EN2**: DAC channel2 enable  
This bit is set and cleared by software to enable/disable DAC channel2.  
0: DAC channel2 disabled  
1: DAC channel2 enabled  
*Note:* *These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*
- Bit 15 Reserved, must be kept at reset value.
- Bit 14 **CEN1**: DAC channel1 calibration enable  
This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC\_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.  
0: DAC channel1 in normal operating mode  
1: DAC channel1 in calibration mode
- Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable  
This bit is set and cleared by software.  
0: DAC channel1 DMA Underrun Interrupt disabled  
1: DAC channel1 DMA Underrun Interrupt enabled
- Bit 12 **DMAEN1**: DAC channel1 DMA enable  
This bit is set and cleared by software.  
0: DAC channel1 DMA mode disabled  
1: DAC channel1 DMA mode enabled
- Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector  
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.  
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1  
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3  
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7  
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15  
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31  
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63  
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127  
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255  
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511  
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023  
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047  
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

- These bits are set and cleared by software.
- 00: wave generation disabled
- 01: Noise wave generation enabled
- 1x: Triangle wave generation enabled
- Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection

- These bits select the external event used to trigger DAC channel1
- 0000: SWTRIG1
- 0001: dac\_ch1\_trg1
- 0010: dac\_ch1\_trg2
- ...
- 1111: dac\_ch1\_trg15

Refer to the trigger selection tables in [Section 16.4.2: DAC pins and internal signals](#) for details on trigger configuration and mapping.

*Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).*

Bit 1 **TEN1**: DAC channel1 trigger enable

- This bit is set and cleared by software to enable/disable DAC channel1 trigger.
- 0: DAC channel1 trigger disabled and data written into the DAC\_DHR1 register are transferred one dac\_pclk clock cycle later to the DAC\_DOR1 register
- 1: DAC channel1 trigger enabled and data from the DAC\_DHR1 register are transferred three dac\_pclk clock cycles later to the DAC\_DOR1 register

*Note: When software trigger is selected, the transfer from the DAC\_DHR1 register to the DAC\_DOR1 register takes only one dac\_pclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable

- This bit is set and cleared by software to enable/disable DAC channel1.
- 0: DAC channel1 disabled
- 1: DAC channel1 enabled

### 16.7.2 DAC software trigger register (DAC\_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

*Note: This bit is cleared by hardware (one dac\_pclk clock cycle later) once the DAC\_DHR2 register value has been loaded into the DAC\_DOR2 register.*

*This bit is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

*Note: This bit is cleared by hardware (one dac\_pclk clock cycle later) once the DAC\_DHR1 register value has been loaded into the DAC\_DOR1 register.*

### 16.7.3 DAC channel1 12-bit right-aligned data holding register (DAC\_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACD1DHR[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 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

### 16.7.4 DAC channel1 12-bit left aligned data holding register (DAC\_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACD1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data  
 These bits are written by software.  
 They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

### 16.7.5 DAC channel1 8-bit right aligned data holding register (DAC\_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data  
 These bits are written by software. They specify 8-bit data for DAC channel1.

### 16.7.6 DAC channel2 12-bit right aligned data holding register (DAC\_DHR12R2)

This register is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[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 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data  
 These bits are written by software. They specify 12-bit data for DAC channel2.

### 16.7.7 DAC channel2 12-bit left aligned data holding register (DAC\_DHR12L2)

This register is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

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

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

### 16.7.8 DAC channel2 8-bit right-aligned data holding register (DAC\_DHR8R2)

This register is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

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.	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.	DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

### 16.7.9 Dual DAC 12-bit right-aligned data holding register (DAC\_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

### 16.7.10 Dual DAC 12-bit left aligned data holding register (DAC\_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

### 16.7.11 Dual DAC 8-bit right aligned data holding register (DAC\_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel1.

### 16.7.12 DAC channel1 data output register (DAC\_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.				DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.



### 16.7.13 DAC channel2 data output register (DAC\_DOR2)

This register is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

### 16.7.14 DAC status register (DAC\_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_FLAG2	DMAU DR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAU DR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													

- Bit 31 **BWST2**: DAC channel2 busy writing sample time flag  
This bit is systematically set just after sample and hold mode enable. It is set each time the software writes the register DAC\_SHSR2, It is cleared by hardware when the write operation of DAC\_SHSR2 is complete. (It takes about 3 LSI periods of synchronization).  
0: There is no write operation of DAC\_SHSR2 ongoing: DAC\_SHSR2 can be written  
1: There is a write operation of DAC\_SHSR2 ongoing: DAC\_SHSR2 cannot be written  
*Note: This bit is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*
- Bit 30 **CAL\_FLAG2**: DAC channel2 calibration offset status  
This bit is set and cleared by hardware  
0: calibration trimming value is lower than the offset correction value  
1: calibration trimming value is equal or greater than the offset correction value  
*Note: This bit is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*
- Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag  
This bit is set by hardware and cleared by software (by writing it to 1).  
0: No DMA underrun error condition occurred for DAC channel2  
1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).  
*Note: This bit is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).*
- Bit 28 Reserved, must be kept at reset value.
- Bit 27 Reserved, must be kept at reset value.
- Bits 26:16 Reserved, must be kept at reset value.
- Bit 15 **BWST1**: DAC channel1 busy writing sample time flag  
This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC\_SHSR1, It is cleared by hardware when the write operation of DAC\_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).  
0: There is no write operation of DAC\_SHSR1 ongoing: DAC\_SHSR1 can be written  
1: There is a write operation of DAC\_SHSR1 ongoing: DAC\_SHSR1 cannot be written
- Bit 14 **CAL\_FLAG1**: DAC channel1 calibration offset status  
This bit is set and cleared by hardware  
0: calibration trimming value is lower than the offset correction value  
1: calibration trimming value is equal or greater than the offset correction value
- Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag  
This bit is set by hardware and cleared by software (by writing it to 1).  
0: No DMA underrun error condition occurred for DAC channel1  
1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)
- Bit 12 Reserved, must be kept at reset value.
- Bit 11 Reserved, must be kept at reset value.
- Bits 10:0 Reserved, must be kept at reset value.

### 16.7.15 DAC calibration control register (DAC\_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

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.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

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

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

### 16.7.16 DAC mode control register (DAC\_MCR)

Address offset: 0x3C

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.	MODE2[2:0]		
													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.	MODE1[2:0]		
													rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

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

Bits 18:16 **MODE2[2:0]**: DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC\_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

- DAC channel2 in normal mode
  - 000: DAC channel2 is connected to external pin with Buffer enabled
  - 001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled
  - 010: DAC channel2 is connected to external pin with buffer disabled
  - 011: DAC channel2 is connected to on chip peripherals with Buffer disabled
- DAC channel2 in sample and hold mode
  - 100: DAC channel2 is connected to external pin with Buffer enabled
  - 101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled
  - 110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled
  - 111: DAC channel2 is connected to on chip peripherals with Buffer disabled

*Note: This register can be modified only when EN2 = 0.*

*Refer to [Section 16.3: DAC implementation](#) for the availability of DAC channel2.*

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

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

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

Bits 2:0 **MODE1[2:0]**: DAC channel1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC\_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel1 mode:

- DAC channel1 in normal mode
  - 000: DAC channel1 is connected to external pin with Buffer enabled
  - 001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
  - 010: DAC channel1 is connected to external pin with Buffer disabled
  - 011: DAC channel1 is connected to on chip peripherals with Buffer disabled
- DAC channel1 in sample & hold mode
  - 100: DAC channel1 is connected to external pin with Buffer enabled
  - 101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
  - 110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
  - 111: DAC channel1 is connected to on chip peripherals with Buffer disabled

*Note: This register can be modified only when EN1 = 0.*

### 16.7.17 DAC channel1 sample and hold sample time register (DAC\_SHSR1)

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
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE1[9:0]**: DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC\_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note:* It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.

### 16.7.18 DAC channel2 sample and hold sample time register (DAC\_SHSR2)

This register is available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

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.	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.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE2[9:0]**: DAC channel2 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC\_SR register is low, if BWST2 = 1, the write operation is ignored.

*Note:* It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x LSI clock period.

### 16.7.19 DAC sample and hold time register (DAC\_SHHR)



Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]**: DAC channel2 hold time (only valid in sample and hold mode).

Hold time = (THOLD[9:0]) x LSI clock period

These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

Note: This register can be modified only when EN2 = 0.

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

Bits 9:0 **THOLD1[9:0]**: DAC channel1 hold time (only valid in sample and hold mode)

Hold time = (THOLD[9:0]) x LSI clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC\_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

### 16.7.20 DAC sample and hold refresh time register (DAC\_SHRR)

Address offset: 0x4C

Reset value: 0x0001 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.	TREFRESH2[7:0]							
								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.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **TREFRESH2[7:0]**: DAC channel2 refresh time (only valid in sample and hold mode)

Refresh time = (TREFRESH[7:0]) x LSI clock period

These bits are available only on dual-channel DACs. Refer to [Section 16.3: DAC implementation](#).

Note: This register can be modified only when EN2 = 0.

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

Bits 7:0 **TREFRESH1[7:0]**: DAC channel1 refresh time (only valid in sample and hold mode)

Refresh time = (TREFRESH[7:0]) x LSI clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC\_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

### 16.7.21 DAC register map

Table 90 summarizes the DAC registers.

Table 90. DAC 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	DAC_CR	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]			WAVE2[2:0]			TSEL2[3:1]			TSEL2[0]	TEN2	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]			WAVE1[1:0]			TSEL1[3:1]			TSEL1[0]	TEN1	EN1	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	DAC_SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																														0	0		
0x08	DAC_DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x0C	DAC_DHR12L1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																										0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	Res.											Res.	Res.	Res.	Res.	DACC2DHR[11:0]													
	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	
0x18	DAC_DHR12L2	Res.											Res.	Res.	Res.	Res.	DACC2DHR[11:0]										Res.	Res.	Res.	Res.				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																	
0x20	DAC_DHR12RD	Res.	Res.	Res.	DACC2DHR[11:0]											Res.	Res.	Res.	Res.	DACC1DHR[11:0]														
	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
0x24	DAC_DHR12LD	DACC2DHR[11:0]											Res.	Res.	Res.	Res.	DACC1DHR[11:0]										Res.	Res.	Res.	Res.				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 90. DAC 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	DAC_DHR8RD	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DAC2DHR[7:0]					DAC1DHR[7:0]										
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DAC1DOR[11:0]															
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res.										DAC2DOR[11:0]					
	Reset value																																
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0	0	0													
0x38	DAC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4]	OTRIM2[3]	OTRIM2[2]	OTRIM2[1]	OTRIM2[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value													X	X	X	X	X															
0x3C	DAC_MCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x40	DAC_SHSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x44	DAC_SHSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x48	DAC_SHHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x4C	DAC_SHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																
0x50-0x54	Reserved	Res.																															
0x58-0x60	Reserved	Res.																															
0x64-0x68	Reserved	Res.																															

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



## 17 Voltage reference buffer (VREFBUF)

### 17.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADC and DAC, and also as voltage reference for external components through the VREF+ pin. When the VREF+ pin is double-bonded with VDDA pin in a package, the voltage reference buffer is not available and must be kept disabled (refer to datasheet for packages pinout description).

### 17.2 VREFBUF functional description

The internal voltage reference buffer supports two voltages<sup>(a)</sup>, which are configured with VRS bits in the VREFBUF\_CSR register:

- VRS = 0: V<sub>REF\_OUT1</sub> around 2.048 V
- VRS = 1: V<sub>REF\_OUT2</sub> around 2.5 V

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

**Table 91. VREF buffer modes**

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V <sub>REF+</sub> pin pulled-down to V <sub>SSA</sub>
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V <sub>REF+</sub> pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V <sub>REF+</sub> pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF\_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

a. The minimum V<sub>DDA</sub> voltage depends on VRS setting, refer to the product datasheet.

## 17.3 VREFBUF registers

### 17.3.1 VREFBUF control and status register (VREFBUF\_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	VRR	VRS	HIZ	ENVR
												r	rw	rw	rw

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

Bit 3 **VRR**: Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 **VRS**: Voltage reference scale

This bit selects the value generated by the voltage reference buffer.

0: Voltage reference set to  $V_{REF\_OUT1}$  (around 2.048 V).

1: Voltage reference set to  $V_{REF\_OUT2}$  (around 2.5 V).

Bit 1 **HIZ**: High impedance mode

This bit controls the analog switch to connect or not the  $V_{REF+}$  pin.

0:  $V_{REF+}$  pin is internally connected to the voltage reference buffer output.

1:  $V_{REF+}$  pin is high impedance.

Refer to [Table 91: VREF buffer modes](#) for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR**: Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.

0: Internal voltage reference mode disable (external voltage reference mode).

1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

### 17.3.2 VREFBUF calibration control register (VREFBUF\_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

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.	TRIM[5:0]					
										rw	rw	rw	rw	rw	rw

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

Bits 5:0 **TRIM[5:0]**: Trimming code

These bits are automatically initialized after reset with the trimming value stored in the flash memory during the production test. Writing into these bits allows the tuning of the internal reference buffer voltage.

*Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.*

### 17.3.3 VREFBUF register map

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

**Table 92. VREFBUF 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	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRR	VRS	HIZ	ENVR
	Reset value																												0	0	1	0	
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

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

## 18 Comparator (COMP)

### 18.1 Introduction

The devices embed COMP1, COMP2, and COMP3 (the last on STM32G0B1xx and STM32G0C1xx only) ultra-low-power comparators.

The comparators can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with a PWM output from a timer.

### 18.2 COMP main features

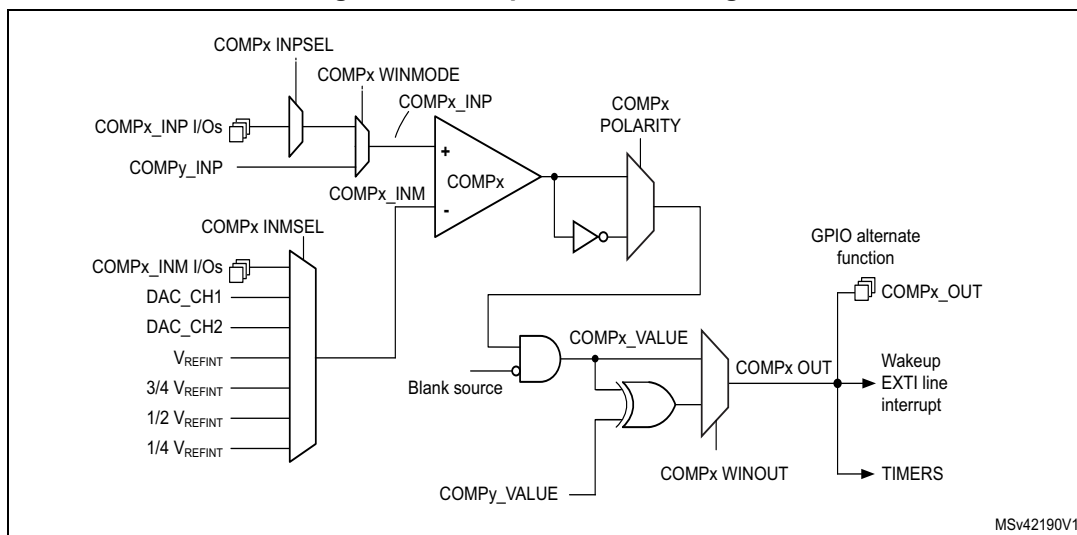
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC Channel1 and Channel2
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Break events for fast PWM shutdowns
- Comparator outputs with blanking source
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wake-up from Sleep and Stop modes (through the EXTI controller)

## 18.3 COMP functional description

### 18.3.1 COMP block diagram

The block diagram of the comparators is shown in [Figure 68](#).

**Figure 68. Comparator block diagram**



### 18.3.2 COMP pins and internal signals

The I/Os used as comparators inputs must be configured in analog mode in the GPIOs registers.

The comparator output can be connected to the I/Os using the alternate function channel given in “Alternate function mapping” table in the datasheet.

The output can also be internally redirected to a variety of timer input for the following purposes:

- Emergency shut-down of PWM signals, using BKIN and BKIN2 inputs
- Cycle-by-cycle current control, using OCREF\_CLR inputs
- Input capture for timing measures

It is possible to have the comparator output simultaneously redirected internally and externally.

**Table 93. COMP1 non-inverting input assignment**

COMP1_INP	COMP1_INPSEL[1:0]
PC5	00
PB2	01
PA1	10
Open	11

**Table 94. COMP1 inverting input assignment**

COMP1_INM	COMP1_INMSEL[3:0]
$\frac{1}{4} V_{REFINT}$	0000
$\frac{1}{2} V_{REFINT}$	0001
$\frac{3}{4} V_{REFINT}$	0010
$V_{REFINT}$	0011
DAC Channel1	0100
DAC Channel2	0101
PB1	0110
PC4	0111
PA0	1000
$\frac{1}{4} V_{REFINT}$	> 1000

**Table 95. COMP2 non-inverting input assignment**

COMP2_INP	COMP2_INPSEL[1:0]
PB4	00
PB6	01
PA3	10
Open	11

**Table 96. COMP2 inverting input assignment**

COMP2_INM	COMP2_INMSEL[3:0]
$\frac{1}{4} V_{REFINT}$	0000
$\frac{1}{2} V_{REFINT}$	0001
$\frac{3}{4} V_{REFINT}$	0010
$V_{REFINT}$	0011
DAC Channel1	0100
DAC Channel2	0101
PB3	0110
PB7	0111
PA2	1000
$\frac{1}{4} V_{REFINT}$	> 1000

Table 97. COMP3 non-inverting input assignment

COMP3_INP	COMP3_INPSEL[1:0]
PB0	00
PC1	01
PE7	10
Open	11

Table 98. COMP3 inverting input assignment

COMP3_INM	COMP3_INMSEL[3:0]
$\frac{1}{4} V_{REFINT}$	0000
$\frac{1}{2} V_{REFINT}$	0001
$\frac{3}{4} V_{REFINT}$	0010
$V_{REFINT}$	0011
DAC Channel1	0100
DAC Channel2	0101
PB2	0110
PC0	0111
PE8	1000
$\frac{1}{4} V_{REFINT}$	> 1000

### 18.3.3 COMP reset and clocks

The COMP clock provided by the clock controller is synchronous with the APB2 clock.

There is no clock enable control bit provided in the RCC controller. Reset and clock enable bits are common for COMP and SYSCFG.

**Important:** The polarity selection logic and the output redirection to the port works independently from the APB2 clock. This allows the comparator to work even in Stop mode.

### 18.3.4 Comparator LOCK mechanism

The comparators can be used for safety purposes, such as over-current or thermal protection. For applications having specific functional safety requirements, it is necessary to insure that the comparator programming cannot be altered in case of spurious register access or program counter corruption.

For this purpose, the comparator control and status registers can be write-protected (read-only).

Once the programming is completed, the COMPx LOCK bit can be set to 1. This causes the whole register to become read-only, including the COMPx LOCK bit.

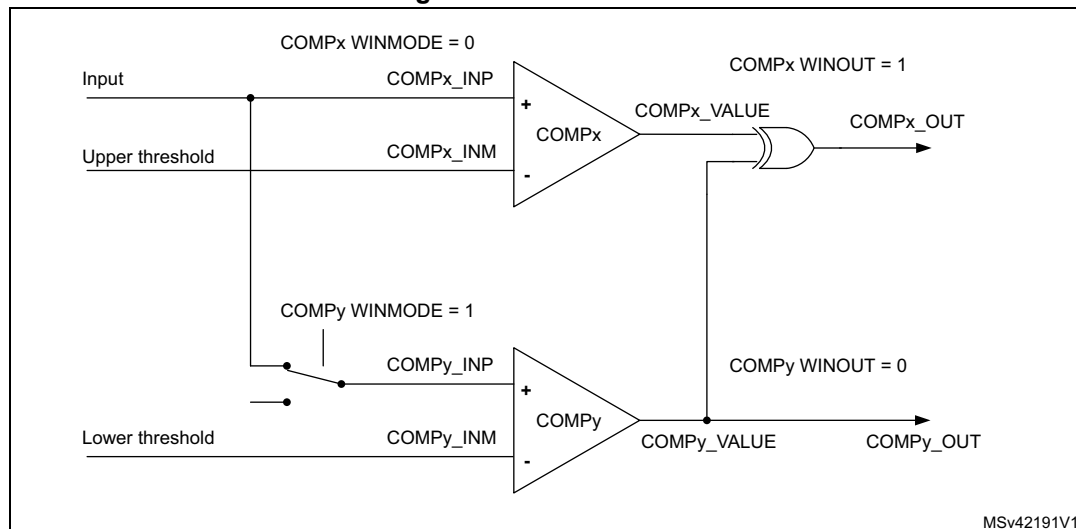
The write protection can only be reset by a MCU reset.

### 18.3.5 Window comparator

The purpose of window comparator is to monitor the analog voltage if it is within specified voltage range defined by lower and upper threshold.

COMP1 and COMP2 or COMP2 and COMP3 can combine to create a window comparator. The monitored analog voltage is connected to the non-inverting (plus) inputs of comparators connected together and the upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators. Two non-inverting inputs can be connected internally together by enabling WINMODE bit to save one IO for other purposes.

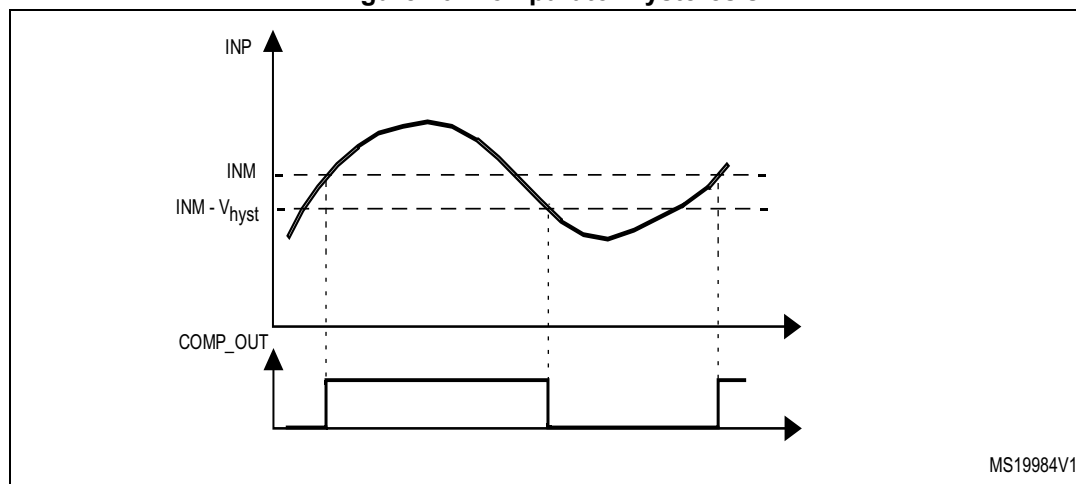
Figure 69. Window mode



### 18.3.6 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if it is not needed (for instance when exiting from low-power mode) to be able to force the hysteresis value using external components.

Figure 70. Comparator hysteresis

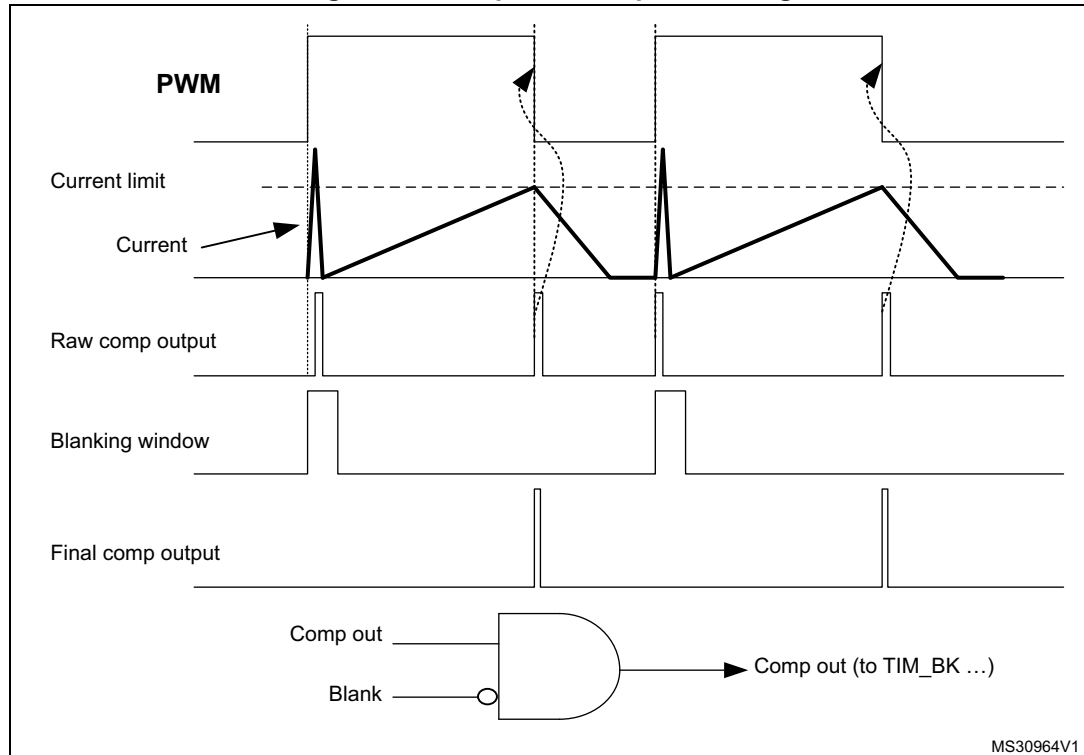




### 18.3.7 Comparator output blanking function

The purpose of the blanking function is to prevent the current regulation to trip upon short current spikes at the beginning of the PWM period (typically the recovery current in power switches anti parallel diodes). It consists of a selection of a blanking window which is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals). Then, the complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output. See the example provided in the figure below.

Figure 71. Comparator output blanking



MS30964V1

### 18.3.8 COMP power and speed modes

COMPx power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

The PWRMODE[1:0] bitfields of the COMPx\_CSR registers allow setting the comparators to high speed with full power or medium speed with medium power. Refer to [Section 18.6: COMP registers](#).

## 18.4 COMP low-power modes

**Table 99. Comparator behavior in the low power modes**

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. COMP interrupts cause the device to exit the Low-power sleep mode.
Stop 0	No effect on the comparators.
Stop 1	Comparator interrupts cause the device to exit the Stop mode.
Standby	The COMP registers are powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

## 18.5 COMP interrupts

The comparator outputs are internally connected to the Extended interrupts and events controller. Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from low-power modes.

Refer to Interrupt and events section for more details.

To enable COMPx interrupt, it is required to follow this sequence:

1. Configure and enable the EXTI line corresponding to the COMPx output event in interrupt mode and select the rising, falling or both edges sensitivity
2. Configure and enable the NVIC IRQ channel mapped to the corresponding EXTI lines
3. Enable COMPx.

**Table 100. Interrupt control bits**

Interrupt event	Enable control bit	Exit from Sleep mode	Exit from Stop modes	Exit from Standby mode
COMP1 output	Through EXTI	Yes	Yes	N/A
COMP2 output	Through EXTI	Yes	Yes	N/A
COMP3 output	Through EXTI	Yes	Yes	N/A

## 18.6 COMP registers

### 18.6.1 Comparator 1 control and status register (COMP1\_CSR)

Address offset: 0x00

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]				PWRMODE [1:0]		HYST[1:0]		
rw	r						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
POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw			rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 **LOCK**: COMP1\_CSR register lock

This bit is set by software and cleared by a system reset. It locks the comparator 3 control bits. When locked, all register bits are read-only.

- 0: Not locked
- 1: Locked

Bit 30 **VALUE**: Comparator 1 output status

This bit is read-only. It reflects the level of the comparator 1 output after the polarity selector and blanking, as indicated in [Figure 68](#).

Bits 29:25 Reserved, must be kept at reset value.

Bits 24:20 **BLANKSEL[4:0]**: Comparator 1 blanking source selector

This bitfield is controlled by software (if not locked). It selects the blanking source:

- 00000: None (no blanking)
- xxxx1: TIM1 OC4
- xxx1x: TIM1 OC5
- xx1xx: TIM2 OC3
- x1xxx: TIM3 OC3
- 1xxxx: TIM15 OC2

Bits 19:18 **PWRMODE[1:0]**: Comparator 1 power mode selector

This bitfield is controlled by software (if not locked). It selects the power consumption and as a consequence the speed of the comparator 1:

- 00: High speed
- 01: Medium speed
- others: Reserved

Bits 17:16 **HYST[1:0]**: Comparator 1 hysteresis selector

This bitfield is controlled by software (if not locked). It selects the hysteresis of the comparator 1:

- 00: None
- 01: Low
- 10: Medium
- 11: High

Bit 15 **POLARITY**: Comparator 1 polarity selector

This bit is controlled by software (if not locked). It selects the comparator 1 output polarity:

- 0: Non-inverted
- 1: Inverted

Bit 14 **WINOUT**: Comparator 1 output selector

This bit is controlled by software (if not locked). It selects the comparator 1 output:

- 0: COMP1\_VALUE
- 1: COMP1\_VALUE XOR COMP2\_VALUE (required for window mode, see [Figure 69](#))

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE**: Comparator 1 non-inverting input selector for window mode

This bit is controlled by software (if not locked). It selects the signal for COMP1\_INP input of the comparator 1:

0: Signal selected with INPSEL[1:0] bitfield of this register

1: COMP2\_INP signal of the comparator 2 (required for window mode, see [Figure 69](#))

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: Comparator 1 signal selector for non-inverting input

This bitfield is controlled by software (if not locked). It selects the signal for the non-inverting input COMP1\_INP of the comparator 1 (also see the WINMODE bit):

00: PC5

01: PB2

10: PA1

11: None (open)

Bits 7:4 **INMSEL[3:0]**: Comparator 1 signal selector for inverting input INM

This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP1\_INM of the comparator 1:

0000: 1/4 V<sub>REFINT</sub>

0001: 1/2 V<sub>REFINT</sub>

0010: 3/4 V<sub>REFINT</sub>

0011: V<sub>REFINT</sub>

0100: DAC channel 1

0101: DAC channel 2

0110: PB1

0111: PC4

1000: PA0

Other: 1/4 V<sub>REFINT</sub>

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Comparator 1 enable bit

This bit is controlled by software (if not locked). It enables the comparator 1:

0: Disable

1: Enable

### 18.6.2 Comparator 2 control and status register (COMP2\_CSR)

Address offset: 0x04

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE [1:0]		HYST[1:0]	
rw	r						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
POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw			rw		rw	rw	rw	rw	rw	rw				rw

- Bit 31 **LOCK**: COMP2\_CSR register lock  
This bit is set by software and cleared by a system reset. It locks the comparator 3 control bits. When locked, all register bits are read-only.  
0: Not locked  
1: Locked
- Bit 30 **VALUE**: Comparator 2 output status  
This bit is read-only. It reflects the level of the comparator 2 output after the polarity selector and blanking, as indicated in [Figure 68](#).
- Bits 29:25 Reserved, must be kept at reset value.
- Bits 24:20 **BLANKSEL[4:0]**: Comparator 2 blanking source selector  
This bitfield is controlled by software (if not locked). It selects the blanking source:  
00000: None (no blanking)  
xxx1x: TIM1 OC4  
xxx1x: TIM1 OC5  
xx1xx: TIM2 OC3  
x1xxx: TIM3 OC3  
1xxxx: TIM15 OC2
- Bits 19:18 **PWRMODE[1:0]**: Comparator 2 power mode selector  
This bitfield is controlled by software (if not locked). It selects the power consumption and as a consequence the speed of the comparator 2:  
00: High speed  
01: Medium speed  
others: Reserved
- Bits 17:16 **HYST[1:0]**: Comparator 2 hysteresis selector  
This bitfield is controlled by software (if not locked). It selects the hysteresis of the comparator 2:  
00: None  
01: Low  
10: Medium  
11: High
- Bit 15 **POLARITY**: Comparator 2 polarity selector  
This bit is controlled by software (if not locked). It selects the comparator 2 output polarity:  
0: Non-inverted  
1: Inverted
- Bit 14 **WINOUT**: Comparator 2 output selector  
This bit is controlled by software (if not locked). It selects the comparator 2 output:  
0: COMP2\_VALUE  
1: COMP1\_VALUE XOR COMP2\_VALUE (required for window mode, see [Figure 69](#))
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WINMODE**: Comparator 2 non-inverting input selector for window mode  
This bit is controlled by software (if not locked). It selects the signal for COMP2\_INP input of the comparator 2:  
0: Signal selected with INPSEL[1:0] bitfield of this register  
1: COMP1\_INP signal of the comparator 1 (required for window mode, see [Figure 69](#))
- Bit 10 Reserved, must be kept at reset value.

- Bits 9:8 **INPSEL[1:0]**: Comparator 2 signal selector for non-inverting input  
 This bitfield is controlled by software (if not locked). It selects the signal for the non-inverting input COMP2\_INP of the comparator 2 (also see the WINMODE bit):  
 00: PB4  
 01: PB6  
 10: PA3  
 11: None (open)
- Bits 7:4 **INMSEL[3:0]**: Comparator 2 signal selector for inverting input INM  
 This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP2\_INM of the comparator 2:  
 0000: 1/4 V<sub>REFINT</sub>  
 0001: 1/2 V<sub>REFINT</sub>  
 0010: 3/4 V<sub>REFINT</sub>  
 0011: V<sub>REFINT</sub>  
 0100: DAC channel 1  
 0101: DAC channel 2  
 0110: PB3  
 0111: PB7  
 1000: PA2  
 Other: 1/4 V<sub>REFINT</sub>
- Bits 3:1 Reserved, must be kept at reset value.
- Bit 0 **EN**: Comparator 2 enable bit  
 This bit is controlled by software (if not locked). It enables the comparator 2:  
 0: Disable  
 1: Enable

### 18.6.3 Comparator 3 control and status register (COMP3\_CSR)

Address offset: 0x08

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]				PWRMODE [1:0]		HYST[1:0]		
rw	r						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
POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw			rw		rw	rw	rw	rw	rw	rw				rw

- Bit 31 **LOCK**: COMP3\_CSR register lock  
 This bit is set by software and cleared by a system reset. It locks the comparator 3 control bits. When locked, all register bits are read-only.  
 0: Not locked  
 1: Locked
- Bit 30 **VALUE**: Comparator 3 output status  
 This bit is read-only. It reflects the level of the comparator 3 output after the polarity selector and blanking, as indicated in [Figure 68](#).
- Bits 29:25 Reserved, must be kept at reset value.

- Bits 24:20 **BLANKSEL[4:0]**: Comparator 3 blanking source selector  
This bitfield is controlled by software (if not locked). It selects the blanking source:  
00000: None (no blanking)  
xxxx1: TIM1 OC4  
xxx1x: TIM1 OC5  
xx1xx: TIM2 OC3  
x1xxx: TIM3 OC3  
1xxxx: TIM15 OC2
- Bits 19:18 **PWRMODE[1:0]**: Comparator 3 power mode selector  
This bitfield is controlled by software (if not locked). It selects the power consumption and as a consequence the speed of the comparator 3:  
00: High speed  
01: Medium speed  
others: Reserved
- Bits 17:16 **HYST[1:0]**: Comparator 3 hysteresis selector  
This bitfield is controlled by software (if not locked). It selects the hysteresis of the comparator 3:  
00: None  
01: Low  
10: Medium  
11: High
- Bit 15 **POLARITY**: Comparator 3 polarity selector  
This bit is controlled by software (if not locked). It selects the comparator 3 output polarity:  
0: Non-inverted  
1: Inverted
- Bit 14 **WINOUT**: Comparator 3 output selector  
This bit is controlled by software (if not locked). It selects the comparator 3 output:  
0: COMP3\_VALUE  
1: COMP2\_VALUE XOR COMP3\_VALUE (required for window mode, see [Figure 69](#))
- Bits 13:12 Reserved, must be kept at reset value.
- Bit 11 **WINMODE**: Comparator 3 non-inverting input selector for window mode  
This bit is controlled by software (if not locked). It selects the signal for COMP3\_INP input of the comparator 3:  
0: Signal selected with INPSEL[1:0] bitfield of this register  
1: COMP2\_INP signal of the comparator 2 (required for window mode, see [Figure 69](#))
- Bit 10 Reserved, must be kept at reset value.
- Bits 9:8 **INPSEL[1:0]**: Comparator 3 signal selector for non-inverting input  
This bitfield is controlled by software (if not locked). It selects the signal for the non-inverting input COMP3\_INP of the comparator 3 (also see the WINMODE bit):  
00: PB0  
01: PC1  
10: PE7  
11: None (open)

Bits 7:4 **INMSEL[3:0]**: Comparator 3 signal selector for inverting input INM

This bitfield is controlled by software (if not locked). It selects the signal for the inverting input COMP3\_INM of the comparator 3:

0000:  $1/4 V_{REFINT}$

0001:  $1/2 V_{REFINT}$

0010:  $3/4 V_{REFINT}$

0011:  $V_{REFINT}$

0100: DAC channel 1

0101: DAC channel 2

0110: PB2

0111: PC0

1000: PE8

Other:  $1/4 V_{REFINT}$

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Comparator 3 enable bit

This bit is controlled by software (if not locked). It enables the comparator 3:

0: Disable

1: Enable



### 18.6.4 COMP register map

The following table summarizes the comparator registers.

The comparator registers share SYSCFG peripheral register base addresses.

**Table 101. COMP register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	COMP1_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]				PWRMODE[1:0]		HYST	POL	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]			Res.	Res.	Res.	EN			
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0
0x04	COMP2_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]				PWRMODE[1:0]		HYST	POL	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]			Res.	Res.	Res.	EN			
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	
0x08	COMP3_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]				PWRMODE[1:0]		HYST	POL	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]			Res.	Res.	Res.	EN			
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

## 19 True random number generator (RNG)

### 19.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG can be used to construct a NIST compliant deterministic random bit generator (DRBG), acting as a live entropy source.

The RNG true random number generator has been tested using the German BSI statistical tests of AIS-31 (T0 to T8).

### 19.2 RNG main features

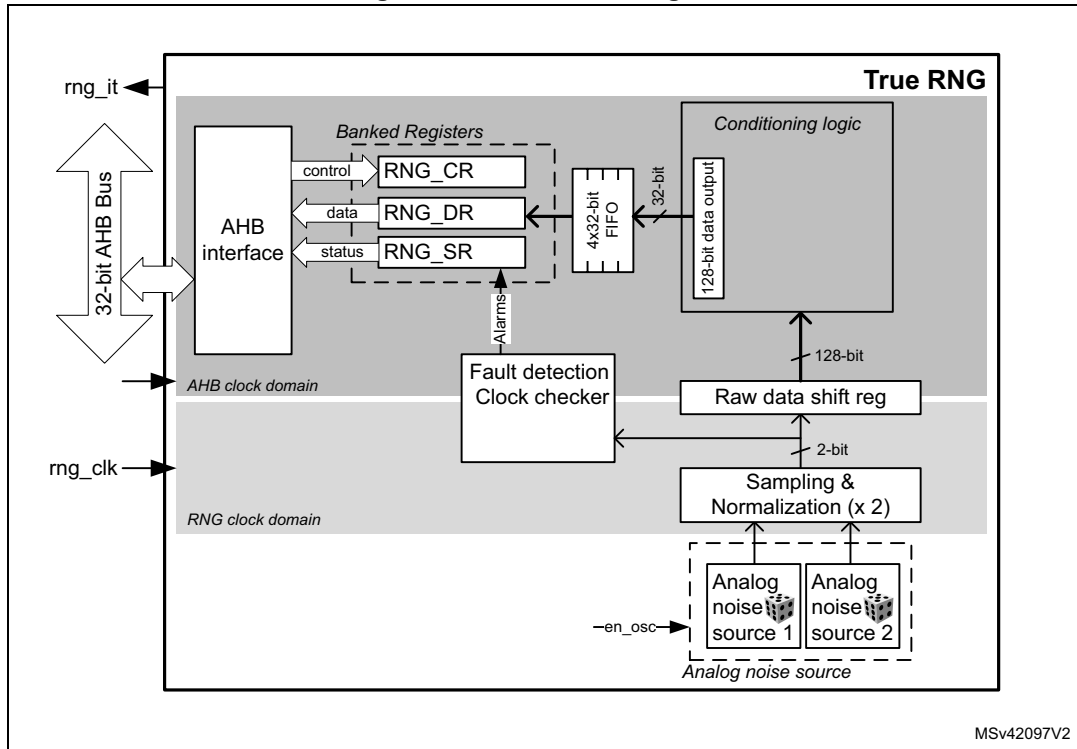
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- In the NIST configuration, it produces four 32-bit random samples every  $16 \times \frac{f_{\text{AHB}}}{f_{\text{RNG}}}$  AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management
  - Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- It has an AMBA<sup>®</sup> AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

### 19.3 RNG functional description

#### 19.3.1 RNG block diagram

Figure 72 shows the RNG block diagram.

Figure 72. RNG block diagram



#### 19.3.2 RNG internal signals

Table 102 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 102. RNG internal input/output signals

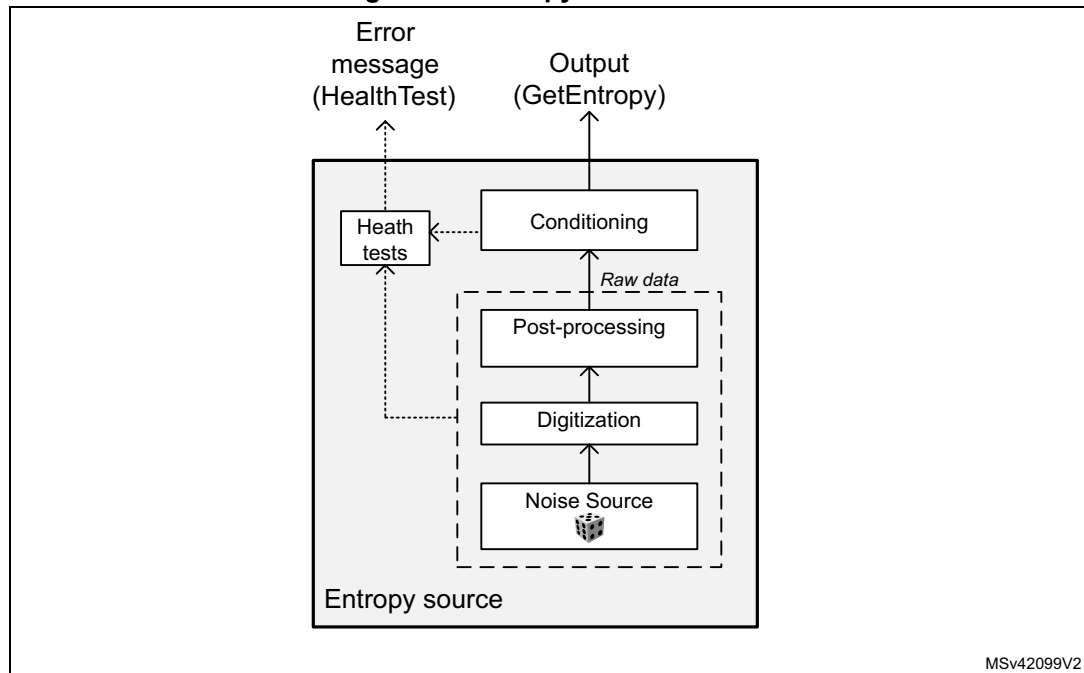
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk

### 19.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. Within its boundary the RNG implements the entropy source model pictured on [Figure 73](#).

It includes an analog noise source, a digitization stage with post-processing, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

**Figure 73. Entropy source model**



The components pictured above are detailed hereafter.

#### Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. It is composed of:

- Two analog noise sources, each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in [Section 19.3.8](#).
- A sampling stage of these outputs clocked by a dedicated clock input (rng\_clk), delivering a 2-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng\_hclk).

*Note:* In [Section 19.6](#) the recommended RNG clock frequencies are given.

#### Post processing

The sample values obtained from a true random noise source consist of 2-bit bitstrings. Because this noise source output is biased, the RNG implements a post-processing component that reduces that bias to a tolerable level.

More specifically, for each of the two noise source bits the RNG takes half of the bits from the sampled noise source, and half of the bits from the inverted sampled noise source. Thus, if the source generates more '1' than '0' (or the opposite), it is filtered.

### Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit).

Also note that post-processing computations are triggered when at least 32 bits of raw datum is received and when output FIFO needs a refill. Thus, the RNG output entropy is maximum when the RNG 128-bit FIFO is emptied by application after 64 RNG clock cycles.

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in [Section 19.5](#).

The conditioning component is clocked by the faster AHB clock.

### Output buffer

A data output buffer can store up to four 32-bit words, which have been output from the conditioning component. When four words have been read from the output FIFO through the RNG\_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register 213 AHB clock cycles later.

Whenever a random number is available through the RNG\_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG\_DR register.

*Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.*

### Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features.

1. Continuous health tests, running indefinitely on the output of the noise source
  - Repetition count test, flagging an error when:
    - a) One of the noise source has provided more than 64 consecutive bits at a constant value ("0" or "1"), or more than 32 consecutive occurrence of two bits patterns ("01" or "10")
    - b) Both noise sources have delivered more than 32 consecutive bits at a constant value ("0" or "1"), or more than 16 consecutive occurrence of two bits patterns ("01" or "10")
2. Vendor specific continuous test
  - Real-time "too slow" sampling clock detector, flagging an error when one RNG clock cycle is smaller than AHB clock cycle divided by 32.

The CECS and SECS status bits in the RNG\_SR register indicate when an error condition is detected, as detailed in [Section 19.3.7: Error management](#).

*Note: An interrupt can be generated when an error is detected.*

### 19.3.4 RNG initialization

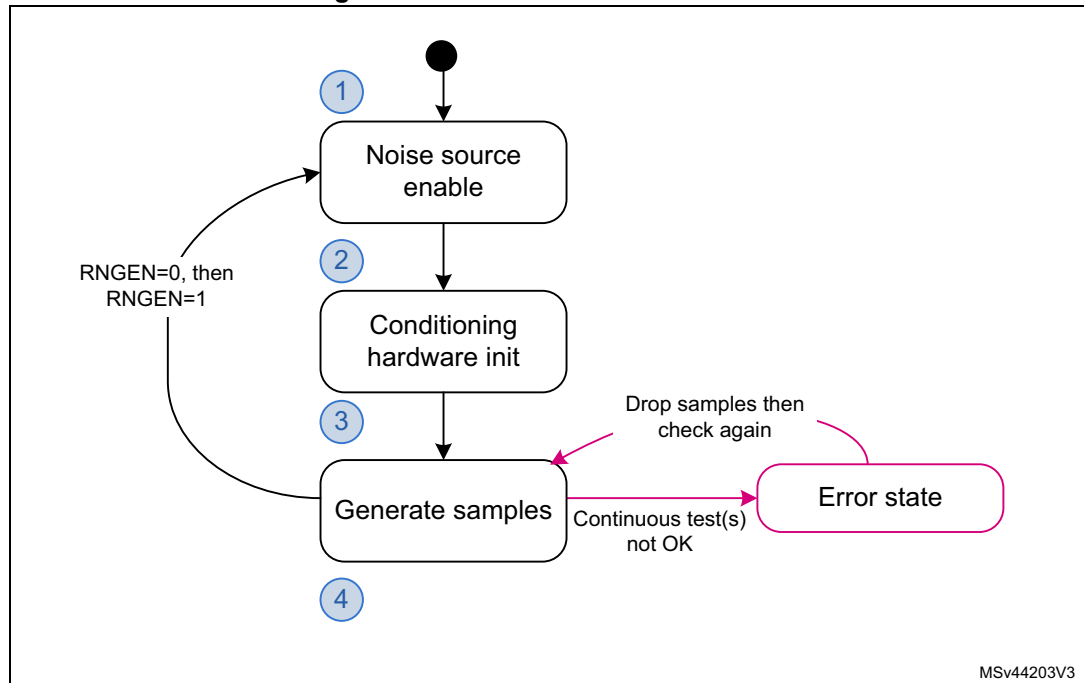
The RNG simplified state machine is pictured on [Figure 74](#).

After enabling the RNG (RNGEN = 1 in RNG\_CR), the following chain of events occurs:

1. The analog noise source is enabled, and logic immediately starts sampling the analog output, filling the 128-bit conditioning shift register.
2. The conditioning logic is enabled and the post-processing context is initialized using two 128 noise source bits.
3. The conditioning stage internal input data buffer is filled again with 128-bit and one conditioning round is performed. The output buffer is then filled with the post processing result.
4. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in [Section 19.5: RNG processing time](#).

**Figure 74. RNG initialization overview**



MSv44203V3

## 19.3.5 RNG operation

### Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Enable the interrupts by setting the IE bit in the RNG\_CR register. At the same time, enable the RNG by setting the bit RNGEN=1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
  - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG\_SR register.
  - A random number is ready. The DRDY bit must be set to 1 in the RNG\_SR register.
  - If the above two conditions are true the content of the RNG\_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG\_DR register must not be read. If an error occurred, the error recovery sequence described in [Section 19.3.7](#) must be used.

To run the RNG in polling mode following steps are recommended:

1. Enable the random number generation by setting the RNGEN bit to “1” in the RNG\_CR register.
2. Read the RNG\_SR register and check that:
  - No error occurred (the SEIS and CEIS bits must be set to 0)
  - A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG\_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG\_DR register must not be read. If an error occurred, the error recovery sequence described in [Section 19.3.7](#) must be used.

*Note:* When data is not ready (DRDY = 0) RNG\_DR returns zero. It is recommended to always verify that RNG\_DR is different from zero. Because when it is the case a seed error occurred between RNG\_SR polling and RND\_DR output reading (rare event).

### Low-power operations

If the power consumption is a concern to the application, low-power strategies can be used, as described in [Section 19.3.8: RNG low-power use](#).

### Software post-processing

If a NIST approved DRBG with 128 bits of security strength is required an approved random generator software must be built around the RNG true random number generator.

Built-in health check functions are described in [Section 19.3.3: Random number generation](#).

### 19.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock is used for noise source sampling. Recommended clock configurations are detailed in [Section 19.6: RNG entropy source validation](#).

*Note:* When the CED bit in the RNG\_CR register is set to 0, the RNG clock frequency the **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG\_SR register).

See [Section 19.3.1: RNG block diagram](#) for details (AHB and RNG clock domains).

### 19.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

#### Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see [Section 19.3.6: RNG clocking](#)) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

*Note:* The clock error has no impact on generated random numbers that is the application can still read the RNG\_DR register.

*CEIS is set only when CECS is set to 1 by RNG.*

#### Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG\_DR register, it must not be used as it may not have enough entropy. If the error was detected during the initialization phase the whole initialization sequence is automatically restarted by the RNG.

The following sequence must be used to fully recover from a seed error after the RNG initialization:

1. Clear the SEIS bit by writing it to "0".
2. Read out 12 words from the RNG\_DR register, and discard each of them in order to clean the pipeline.
3. Confirm that SEIS is still cleared. Random number generation is back to normal.



### 19.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG\_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to the software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG\_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG\_DR register. If it is not the case the RNG must be re-enabled by the application until at least 32 new bits are collected from the noise source and a complete conditioning round is done. It corresponds to 16 RNG clock cycles to sample new bits, and 216 AHB clock cycles to run a conditioning round.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG\_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1.

## 19.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see [Section 19.3.7: Error management](#)
- Clock error, see [Section 19.3.7: Error management](#)

Dedicated interrupt enable control bits are available as shown in [Table 103](#).

**Table 103. RNG interrupt requests**

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write 0 to SEIS
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG\_CR register. The status of the individual interrupt sources can be read from the RNG\_SR register.

*Note:* *Interrupts are generated only when RNG is enabled.*

## 19.5 RNG processing time

The conditioning stage can produce four 32-bit random numbers every  $16 \times \frac{f_{\text{AHB}}}{f_{\text{RNG}}}$  clock cycles, if the value is higher than 213 cycles (213 cycles otherwise). More time is needed for the first set of random numbers after the device exits reset (see

[Section 19.3.4: RNG initialization](#)). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if  $f_{\text{AHB}} < f_{\text{threshold}}$
- 192 RNG clock cycles + 213 AHB cycles, if  $f_{\text{AHB}} \geq f_{\text{threshold}}$

With  $f_{\text{threshold}} = (213 \times f_{\text{RNG}}) / 64$

## 19.6 RNG entropy source validation

### 19.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8). The results can be provided on demand or the customer can reproduce the tests.

### 19.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock  $\text{rng\_clk} = 48 \text{ MHz}$  (CED bit = '0' in RNG\_CR register) and  $\text{rng\_clk} = 400 \text{ kHz}$  (CED bit = '1' in RNG\_CR register).

Table 104. RNG configurations

Configuration	RNG_CR bits						Loop number (N)	RNG_HTCR register	RNG_NCSR register
	NISTC bit	RNG_CONFIG1 [5:0]	CLKDIV [3:0]	RNG_CONFIG2 [2:0] <sup>(1)</sup>	RNG_CONFIG3 [3:0] <sup>(2)</sup>	CED bit			
A	Refer to <i>NIST compliant RNG configuration</i> table in AN4230 available from <a href="http://www.st.com">www.st.com</a> . This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on <a href="https://csrc.nist.gov/projects/cryptographic-module-validation-program">https://csrc.nist.gov/projects/cryptographic-module-validation-program</a> .								
B	1	0x18	0x0 <sup>(3)</sup>	0x0	0x0	0	1	0x0000 NA <sup>(4)</sup>	default
C	0	0x0F		0x0	0xD	0	2		default

1. 0x1 value is recommended instead of 0x0 for RNG\_CONFIG2[2:0], when RNG power consumption is critical. See the end of [Section 19.3.8: RNG low-power use](#) for details.
2. RNG\_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4
3. The noise source sampling must be NA or less. Hence, if the RNG clock is different from NA, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in [Section 19.7.1](#) for details.
4. This value can be fixed in the RNG driver (it doesn't depend on the STM32 family).

### 19.6.3 Data collection

In order to run statistical tests, it is required to collect samples from the entropy source at the raw data level as well as at the output of the entropy source. For details on data collection and the running of statistical test suites refer to "STM32 microcontrollers random number generation validation using NIST statistical test suite" application note (AN4230) available from [www.st.com](http://www.st.com).

*In bypass mode the bits [31:30] of the 4th word are always stuck at zero. Hence the continuous capture of samples is started from the 5th word.*

## 19.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

### 19.7.1 RNG control register (RNG\_CR)

Address offset: 0x000

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.	CED	Res.	IE	RNGEN	Res.	Res.
										rw		rw	rw		

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

Bit 5 **CE**D: Clock error detection

0: Clock error detection enabled

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE**: Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS is set in the RNG\_SR register.

Bit 2 **RNGEN**: True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

### 19.7.2 RNG status register (RNG\_SR)

Address offset: 0x004

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.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

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

**Bit 6 SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0. Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG\_CR register.

**Bit 5 CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct ( $f_{RNGCLK} > f_{HCLK}/32$ )

1: The RNG is detected too slow ( $f_{RNGCLK} < f_{HCLK}/32$ )

An interrupt is pending if IE = 1 in the RNG\_CR register.

Bits 4:3 Reserved, must be kept at reset value.

**Bit 2 SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequences has been detected:

- One of the noise sources has provided more than 64 consecutive bits at a constant value ("0" or "1"), or more than 32 consecutive occurrence of two bit patterns ("01" or "10")
- Both noise sources have delivered more than 32 consecutive bits at a constant value ("0" or "1"), or more than 16 consecutive occurrence of two bit patterns ("01" or "10")

**Bit 1 CECS:** Clock error current status

0: The RNG clock is correct ( $f_{RNGCLK} > f_{HCLK}/32$ ). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ( $f_{RNGCLK} < f_{HCLK}/32$ ).

*Note: CECS bit is valid only if the CED bit in the RNG\_CR register is set to 0.*

**Bit 0 DRDY:** Data ready

0: The RNG\_DR register is not yet valid, no random data is available.

1: The RNG\_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG\_DR register), this bit returns to 0 until a new random value is generated.

*Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG\_CR register).*

If IE=1 in the RNG\_CR register, an interrupt is generated when DRDY = 1.

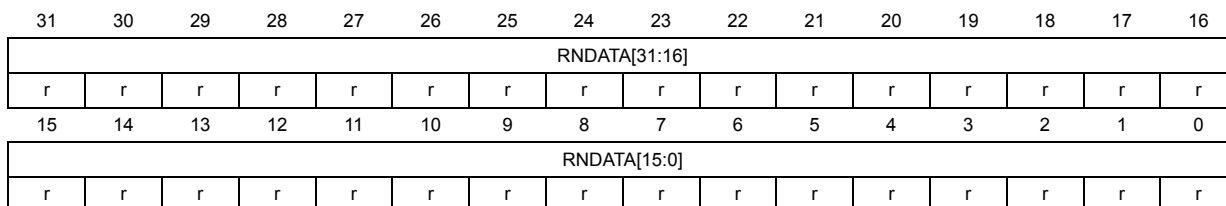
### 19.7.3 RNG data register (RNG\_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG\_DR register is a read-only register that delivers a 32-bit random value when read. After being read, this register delivers a new random value after 216 periods of AHB clock if the output FIFO is empty.

The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.



Bits 31:0 **RNDATA[31:0]**: Random data

32-bit random data, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.

When DRDY is set, it is recommended to always verify that RNG\_DR is different from zero. The zero value means that a seed error occurred between RNG\_SR polling and RND\_DR output reading (a rare event).

### 19.7.4 RNG register map

Table 105. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x000	RNG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	RNG_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.	SEIS	CEIS	Res.	Res.	IE	RNGEN	Res.	Res.
	Reset value																										0	0		0	0	0	0	0
0x008	RNG_DR	RNDATA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to [Section 2.2: Memory organization](#) for the register boundary addresses.

## 20 AES hardware accelerator (AES)

### 20.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data, using an algorithm and implementation fully compliant with the advanced encryption standard (AES) defined in Federal information processing standards (FIPS) publication 197.

The peripheral supports CTR, GCM, GMAC, CCM, ECB, and CBC chaining modes for key sizes of 128 or 256 bits.

AES is an AMBA AHB slave peripheral accessible through 32-bit single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels required).

### 20.2 AES main features

- Compliance with NIST “*Advanced encryption standard (AES), FIPS publication 197*” from November 2001
- 128-bit data block processing
- Support for cipher key lengths of 128-bit and 256-bit
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block of data with, respectively, 128-bit or 256-bit key
- Integrated round key scheduler to compute the last round key for ECB/CBC decryption
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only
- 256-bit write-only register for storing the cryptographic key (eight 32-bit registers)
- 128-bit register for storing initialization vector (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control with support of single-transfer direct memory access (DMA) using two channels (one for incoming data, one for processed data)
- Data-swapping logic to support 1-, 8-, 16- or 32-bit data
- Possibility for software to suspend a message if AES needs to process another message with a higher priority, then resume the original message

### 20.3 AES implementation

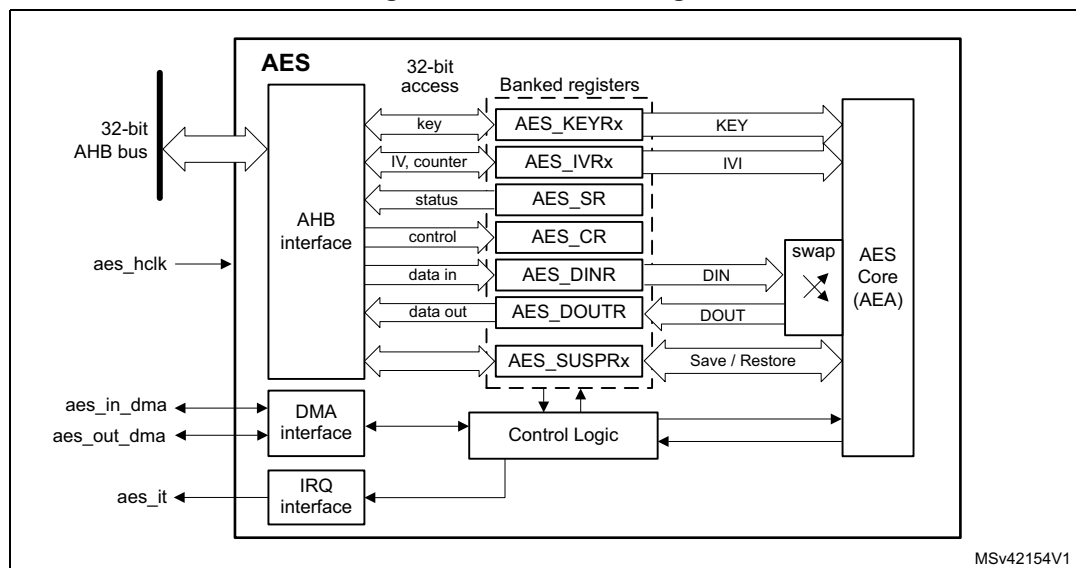
The devices have one AES peripheral.

## 20.4 AES functional description

### 20.4.1 AES block diagram

Figure 75 shows the block diagram of AES.

Figure 75. AES block diagram



### 20.4.2 AES internal signals

Table 106 describes the user relevant internal signals interfacing the AES peripheral.

Table 106. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	Input DMA single request/acknowledge
aes_out_dma	Input/Output	Output DMA single request/acknowledge

### 20.4.3 AES cryptographic core

#### Overview

The AES cryptographic core consists of the following components:

- AES core algorithm (AEA)
- multiplier over a binary Galois field (GF2mul)
- key input
- initialization vector (IV) input
- chaining algorithm logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks (four words) with 128-bit or 256-bit key length. Depending on the chaining mode, the AES requires zero or one 128-bit initialization vector IV.

The AES features the following modes of operation:

- **Mode 1:**  
Plaintext encryption using a key stored in the AES\_KEYRx registers
- **Mode 2:**  
ECB or CBC decryption key preparation. It must be used prior to selecting Mode 3 with ECB or CBC chaining modes. The key prepared for decryption is stored automatically in the AES\_KEYRx registers. Now the AES peripheral is ready to switch to Mode 3 for executing data decryption.
- **Mode 3:**  
Ciphertext decryption using a key stored in the AES\_KEYRx registers. When ECB and CBC chaining modes are selected, the key must be prepared beforehand, through Mode 2.
- **Mode 4:**  
ECB or CBC ciphertext single decryption using the key stored in the AES\_KEYRx registers (the initial key is derived automatically).

*Note: Mode 2 and mode 4 are only used when performing ECB and CBC decryption. When Mode 4 is selected only one decryption can be done, therefore usage of Mode 2 and Mode 3 is recommended instead.*

The operating mode is selected by programming the MODE[1:0] bitfield of the AES\_CR register. It may be done only when the AES peripheral is disabled.

### Typical data processing

Typical usage of the AES is described in [Section 20.4.4: AES procedure to perform a cipher operation on page 473](#).

*Note: The outputs of the intermediate AEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IVI bitfield.*

### Chaining modes

The following chaining modes are supported by AES, selected through the CHMOD[2:0] bitfield of the AES\_CR register:

- Electronic code book (ECB)
- Cipher block chaining (CBC)
- Counter (CTR)
- Galois counter mode (GCM)
- Galois message authentication code (GMAC)
- Counter with CBC-MAC (CCM)

*Note: The chaining mode may be changed only when AES is disabled (bit EN of the AES\_CR register cleared).*

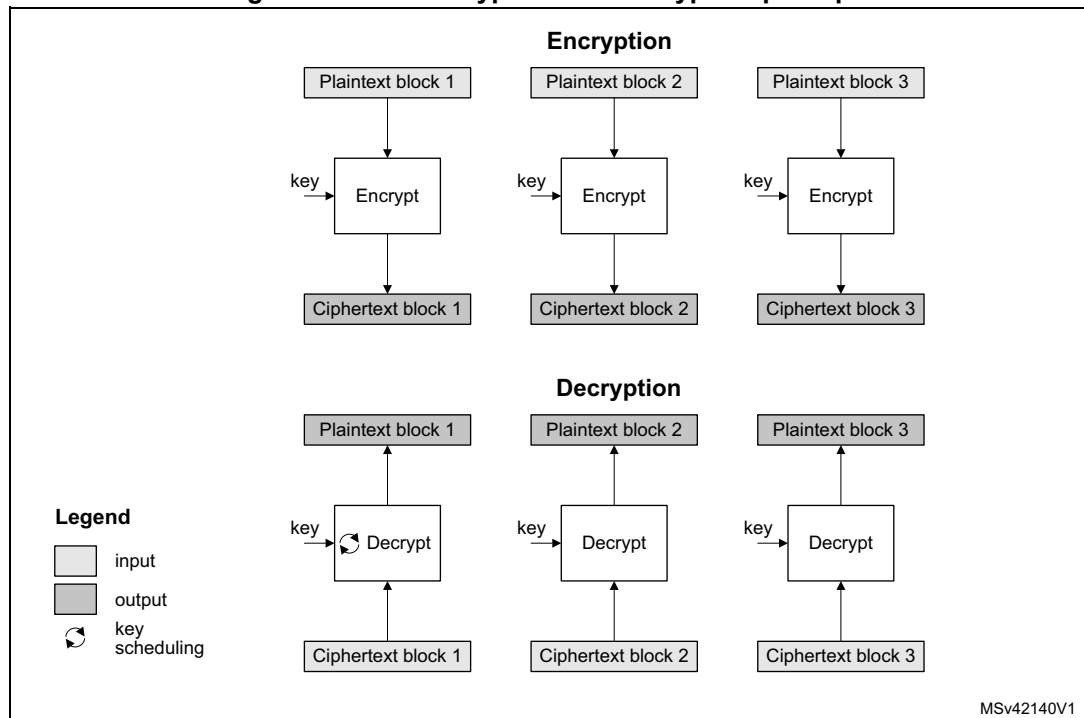
Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from [Section 20.4.8: AES basic chaining modes \(ECB, CBC\)](#).



Electronic codebook (ECB) mode

Figure 76. ECB encryption and decryption principle

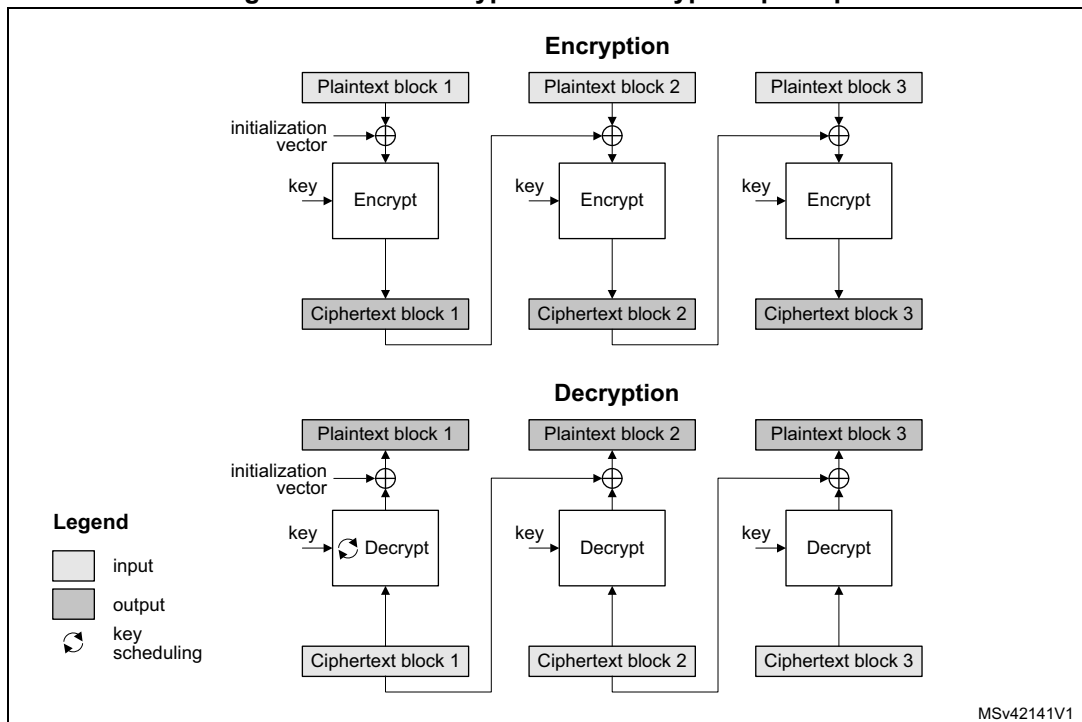


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

*Note:* For decryption, a special key scheduling is required before processing the first block.

### Cipher block chaining (CBC) mode

Figure 77. CBC encryption and decryption principle

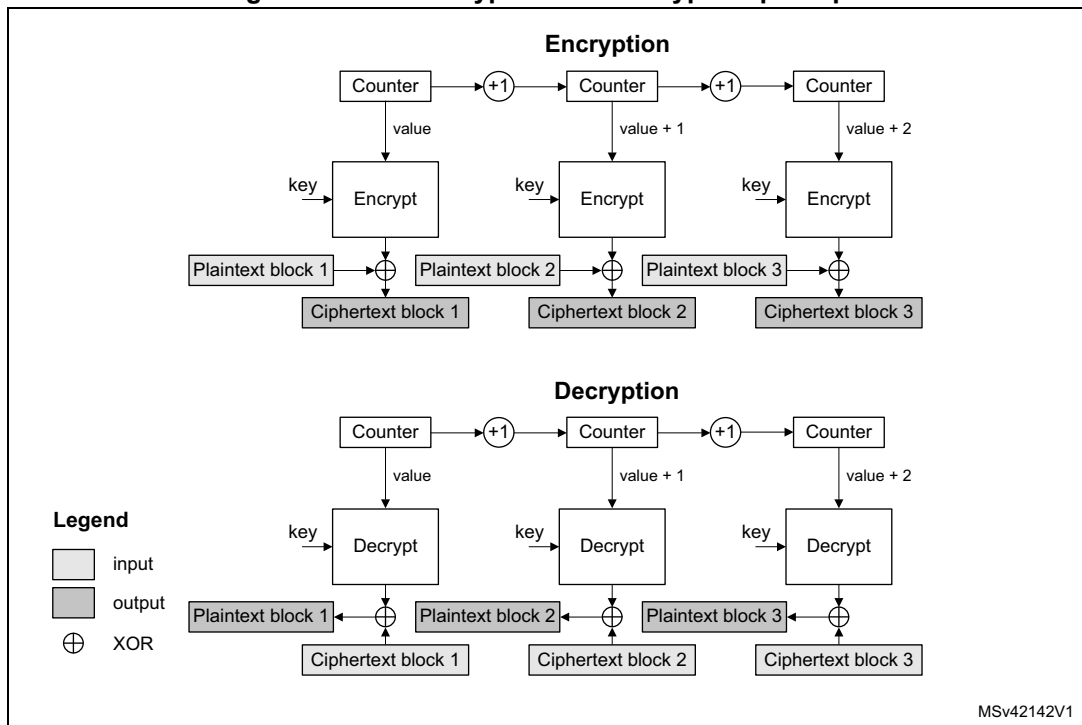


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

*Note:* For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 78. CTR encryption and decryption principle

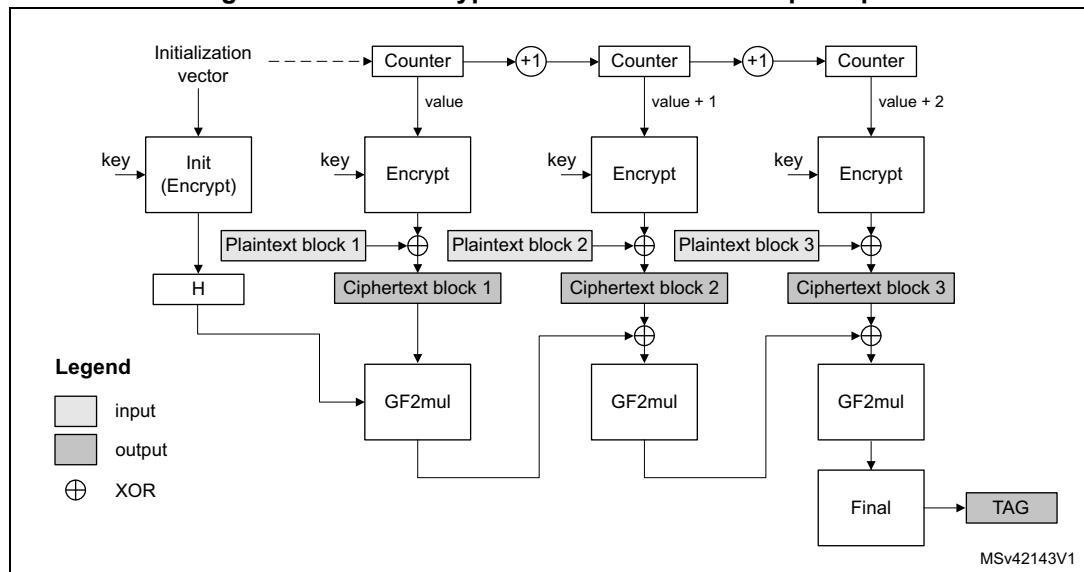


The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

*Note:* Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the key stream, or counter blocks.

Galois/counter mode (GCM)

Figure 79. GCM encryption and authentication principle

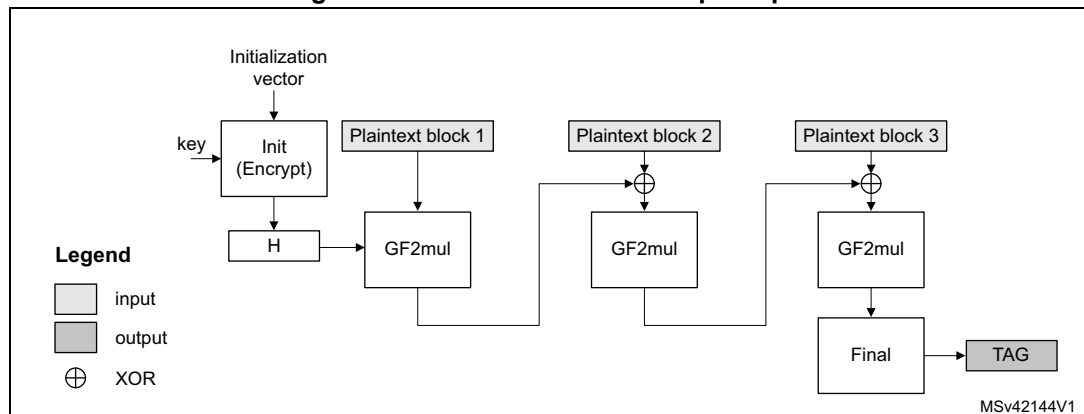


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

Galois message authentication code (GMAC) principle

Figure 80. GMAC authentication principle

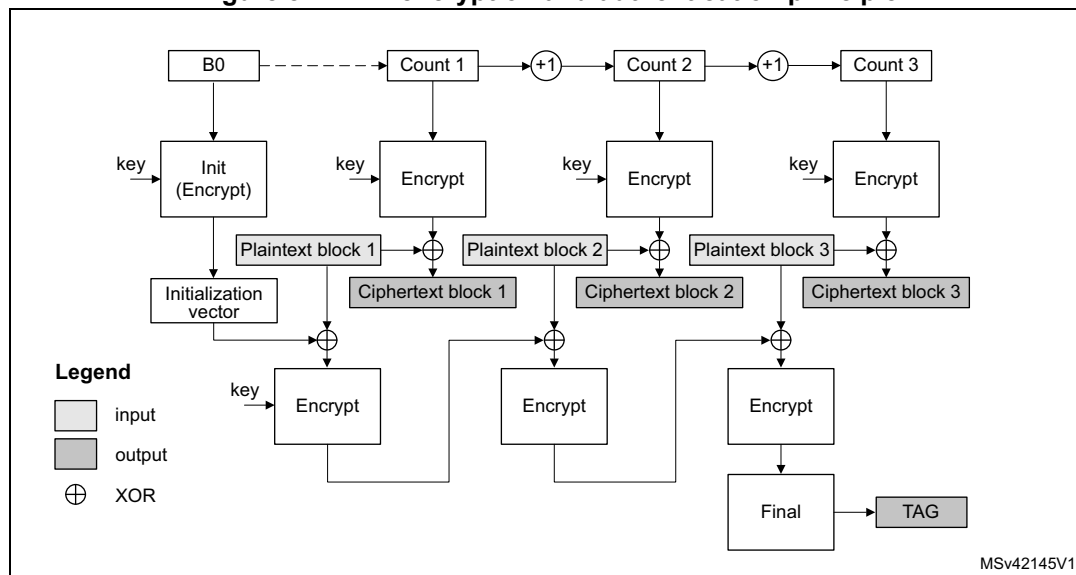


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload).

**Counter with CBC-MAC (CCM) principle**

**Figure 81. CCM encryption and authentication principle**



In Counter with cipher block chaining-message authentication code (CCM) mode, the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM, the CCM chaining mode can be applied on a message composed only by plaintext authenticated data (that is, only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its use is not recommended by NIST.

**20.4.4 AES procedure to perform a cipher operation**

**Introduction**

A typical cipher operation is explained below. Detailed information is provided in sections starting from [Section 20.4.8: AES basic chaining modes \(ECB, CBC\)](#).

## Initialization of AES

To initialize AES, first disable it by clearing the EN bit of the AES\_CR register. Then perform the following steps in any order:

- Configure the AES mode, by programming the MODE[1:0] bitfield of the AES\_CR register.
  - For encryption, select Mode 1 (MODE[1:0] = 00).
  - For decryption, select Mode 3 (MODE[1:0] = 10), unless ECB or CBC chaining modes are used. In this latter case, perform an initial key derivation of the encryption key, as described in [Section 20.4.5: AES decryption round key preparation](#).
- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the AES\_CR register.
- Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE[1:0] bitfield in the AES\_CR register.
- When it is required (for example in CBC or CTR chaining modes), write the initialization vector into the AES\_IVRx registers.
- Configure the key size (128-bit or 256-bit), with the KEYSIZE bitfield of the AES\_CR register.
- Write a symmetric key into the AES\_KEYRx registers (4 or 8 registers depending on the key size).

## Data append

This section describes different ways of appending data for processing, where the size of data to process is not a multiple of 128 bits.

For ECB or CBC mode, refer to [Section 20.4.6: AES ciphertext stealing and data padding](#). The last block management in these cases is more complex than in the sequence described in this section.

### Data append through polling

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
2. Repeat the following sub-sequence until the payload is entirely processed:
  - a) Write four input data words into the AES\_DINR register.
  - b) Wait until the status flag CCF is set in the AES\_SR, then read the four data words from the AES\_DOUTR register.
  - c) Clear the CCF flag, by setting the CCFC bit of the AES\_CR register.
  - d) If the data block just processed is the second-last block of the message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES\_CR register, for AES to compute a correct tag;.
3. As it is the last block, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES\_CR register.

*Note:* Up to three wait cycles are automatically inserted between two consecutive writes to the AES\_DINR register, to allow sending the key to the AES processor.

*NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.*

### Data append using interrupt

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES by setting the CCFIE bit of the AES\_CR register.
2. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. Write first four input data words into the AES\_DINR register.
4. Handle the data in the AES interrupt service routine, upon interrupt:
  - a) Read four output data words from the AES\_DOUTR register.
  - b) Clear the CCF flag and thus the pending interrupt, by setting the CCFC bit of the AES\_CR register.
  - c) If the data block just processed is the second-last block of an message and the significant data in the last block to process is inferior to 128 bits, pad the remainder of the last block with zeros and, in case of GCM payload encryption or CCM payload decryption, specify the number of non-valid bytes, using the NPBLB bitfield of the AES\_CR register, for AES to compute a correct tag;. Then proceed with point 4e).
  - d) If the data block just processed is the last block of the message, discard the data that is not part of the data, then disable the AES peripheral by clearing the EN bit of the AES\_CR register and quit the interrupt service routine.
  - e) Write next four input data words into the AES\_DINR register and quit the interrupt service routine.

*Note:* AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations. NPBLB bits are not used in header phase of GCM, GMAC and CCM chaining modes.

### Data append using DMA

With this method, all the transfers and processing are managed by DMA and AES. To use the method, proceed as follows:

1. Prepare the last four-word data block (if the data to process does not fill it completely), by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in [Section 20.4.16: AES DMA interface](#). Configure the DMA controller so as to generate an interrupt on transfer completion. In case of GCM payload encryption or CCM payload decryption, DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in [Data append through polling](#) must be used instead for this last block, because NPBLB bits must be setup before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral by setting the EN bit of the AES\_CR register
4. Enable DMA requests by setting the DMAINEN and DMAOUTEN bits of the AES\_CR register.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

*Note:* The CCF flag has no use with this method, because the reading of the AES\_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

*NPBLB bits are not used in header phase of GCM, GMAC, and CCM chaining modes.*

### 20.4.5 AES decryption round key preparation

Internal key schedule is used to generate AES round keys. In AES encryption, the round 0 key is the one stored in the key registers. AES decryption must start using the last round key. As the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This key scheduling is only required for AES decryption in ECB and CBC modes.

Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES\_CR (key process only), then proceed with the decryption by setting MODE[1:0] to 10 (Mode 3, decryption only). Mode 2 usage is described below:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select Mode 2 by setting to 01 the MODE[1:0] bitfield of the AES\_CR. The CHMOD[2:0] bitfield is not significant in this case because this key derivation mode is independent of the chaining algorithm selected.
3. Set key length to 128 or 256 bits, via KEYSIZE bit of AES\_CR register.
4. Write the AES\_KEYRx registers (128 or 256 bits) with encryption key. Writes to the AES\_IVRx registers have no effect.
5. Enable the AES peripheral, by setting the EN bit of the AES\_CR register.
6. Wait until the CCF flag is set in the AES\_SR register.
7. Clear the CCF flag. Derived key is available in AES core, ready to use for decryption.

*Note:* The AES is disabled by hardware when the derivation key is available.

*To restart a derivation key computation, repeat steps 4, 5, 6, and 7.*

*Note:* The operation of the key preparation lasts 59 or 82 clock cycles, depending on the key size (128- or 256-bit).

### 20.4.6 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (128 bits), ciphertext stealing techniques are used, such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since the AES peripheral does not support such techniques, the application must complete the last block of input data using data from the second last block.

*Note:* Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when AES is used in other modes than ECB or CBC, an incomplete input data block (that is, block with input data shorter than 128 bits) must be padded with zeros prior to encryption (that is, extra bits must be appended to the trailing end of the data string). After decryption, the extra bits must be discarded. As AES does not implement automatic data padding operation to **the last block**, the application must follow the recommendation given in [Section 20.4.4: AES procedure to perform a cipher operation on page 473](#) to manage messages the size of which is not a multiple of 128 bits.

*Note:* Padding data are swapped in a similar way as normal data, according to the DATATYPE[1:0] field of the AES\_CR register (see [Section 20.4.13: AES data registers and data swapping for details](#)).



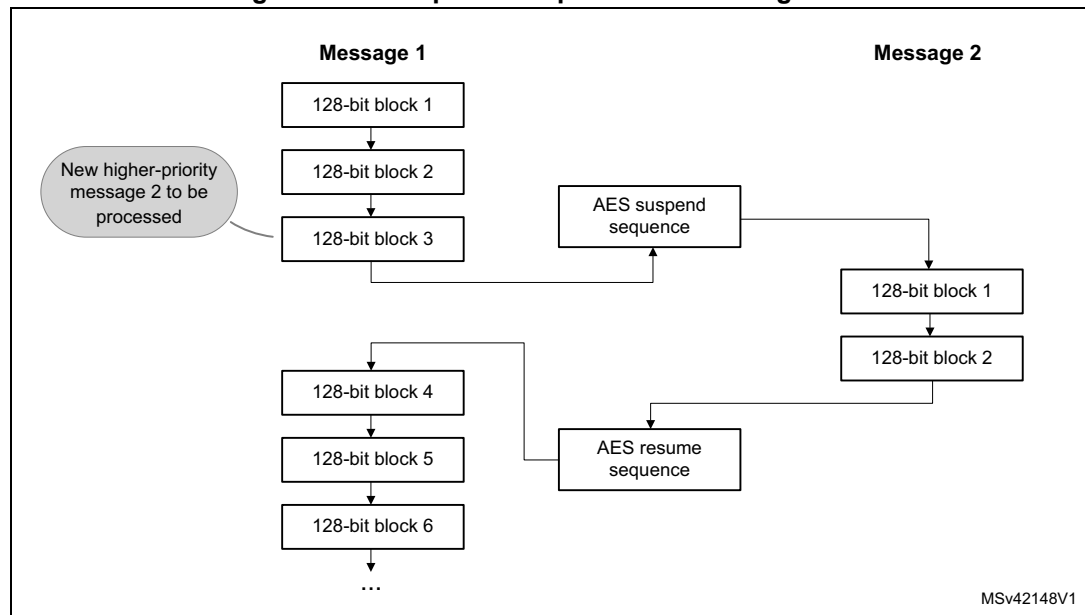
### 20.4.7 AES task suspend and resume

A message can be suspended if another message with a higher priority must be processed. When this highest priority message is sent, the suspended message can resume in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can resume as soon as AES is enabled again to receive the next data block.

Figure 82 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 82. Example of suspend mode management



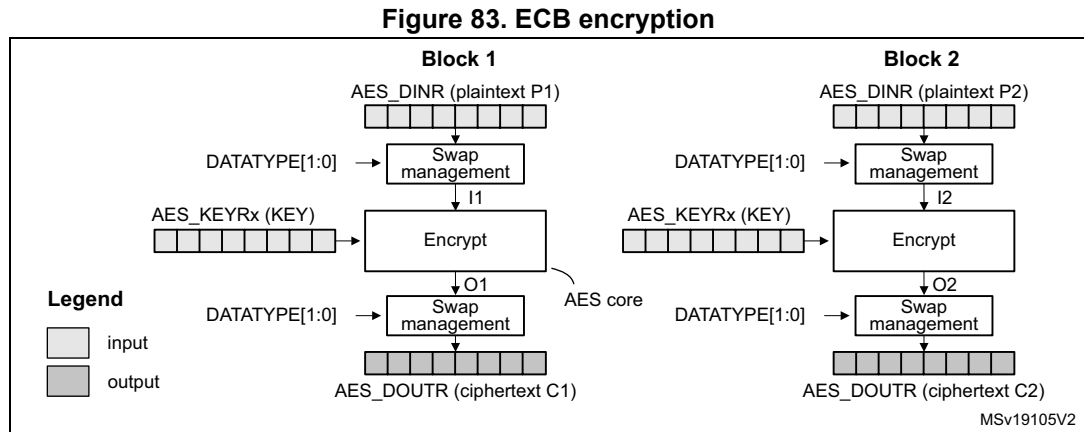
A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

### 20.4.8 AES basic chaining modes (ECB, CBC)

#### Overview

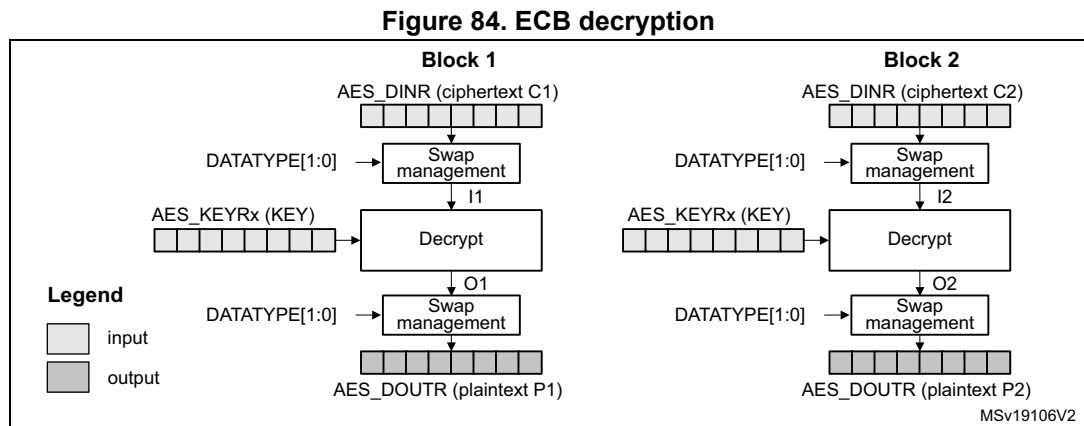
This section gives a brief explanation of the four basic operation modes provided by the AES core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.

Figure 83 illustrates the electronic codebook (ECB) encryption.



In ECB encrypt mode, the 128-bit plaintext input data block P<sub>x</sub> in the AES\_DINR register first goes through bit/byte/half-word swapping. The swap result I<sub>x</sub> is processed with the AES core set in encrypt mode, using a 128- or 256-bit key. The encryption result O<sub>x</sub> goes through bit/byte/half-word swapping, then is stored in the AES\_DOUTR register as 128-bit ciphertext output data block C<sub>x</sub>. The ECB encryption continues in this way until the last complete plaintext block is encrypted.

Figure 84 illustrates the electronic codebook (ECB) decryption.

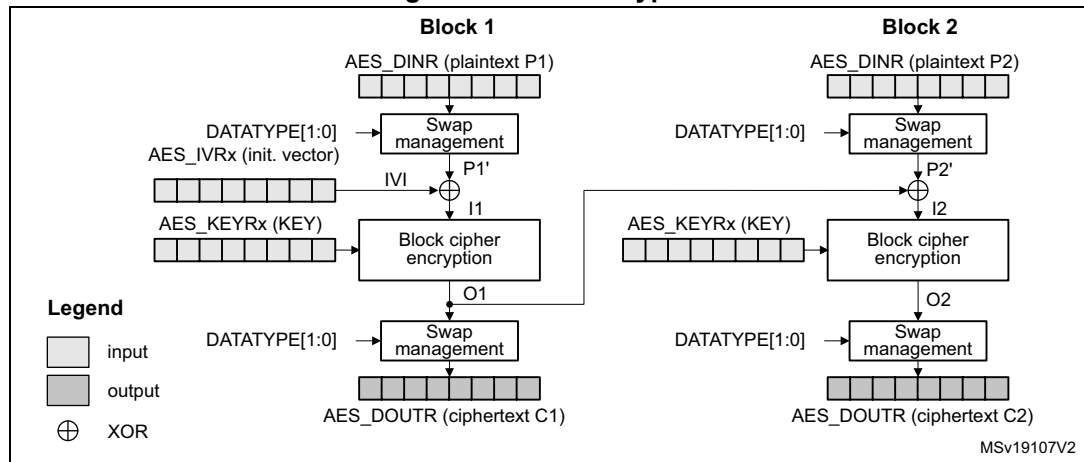


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block C<sub>1</sub> in the AES\_DINR register first goes through bit/byte/half-word swapping. The keying sequence is reversed compared to that of the ECB encryption. The swap result I<sub>1</sub> is processed with the AES core set in decrypt mode, using the formerly prepared decryption key. The decryption result goes through bit/byte/half-word swapping, then is stored in the AES\_DOUTR register as 128-bit plaintext output data block P<sub>1</sub>. The ECB decryption continues in this way until the last complete ciphertext block is decrypted.

Figure 85 illustrates the cipher block chaining (CBC) encryption.

Figure 85. CBC encryption

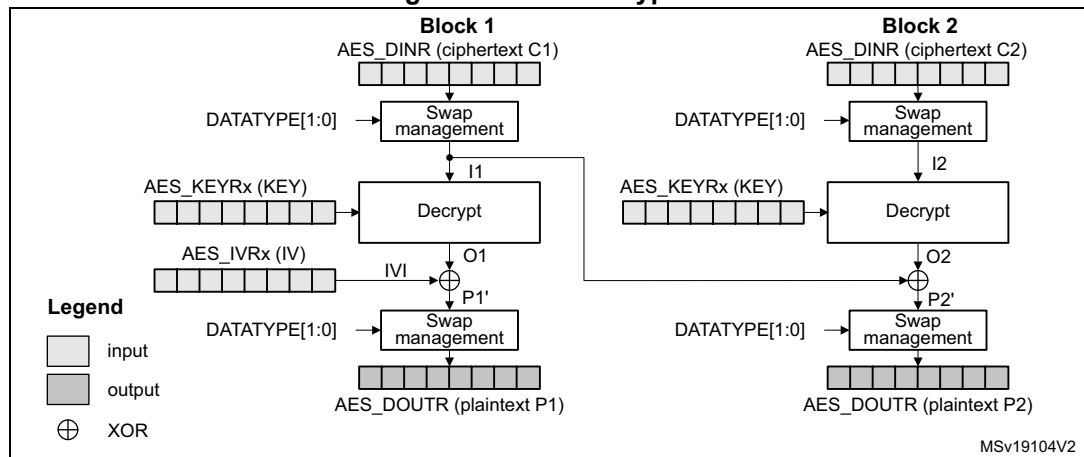


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after swapping operation, is used as ciphertext C1. The O1 data is then XOR-ed with the second-block plaintext data P2' to produce the I2 input data for the AES core to produce the second block of ciphertext data. The chaining of data blocks continues in this way until the last plaintext block in the message is encrypted.

If the message size is not a multiple of 128 bits, the final partial data block is encrypted in the way explained in [Section 20.4.6: AES ciphertext stealing and data padding](#).

Figure 86 illustrates the cipher block chaining (CBC) decryption.

Figure 86. CBC decryption



In CBC decrypt mode, like in ECB decrypt mode, the secret key must be prepared to perform an AES decryption.

After the key preparation process, the decryption goes as follows: the first 128-bit ciphertext block (after the swap operation) is used directly as the AES core input block I1 for decrypt operation, using the 128-bit or 256-bit key. Its output O1 is XOR-ed with the 128-bit IVI field (that must be identical to that used during encryption) to produce the first plaintext block P1.

The second ciphertext block is processed in the same way as the first block, except that the 11 data from the first block is used in place of the initialization vector.

The decryption continues in this way until the last complete ciphertext block is decrypted.

If the message size is not a multiple of 128 bits, the final partial data block is decrypted in the way explained in [Section 20.4.6: AES ciphertext stealing and data padding](#).

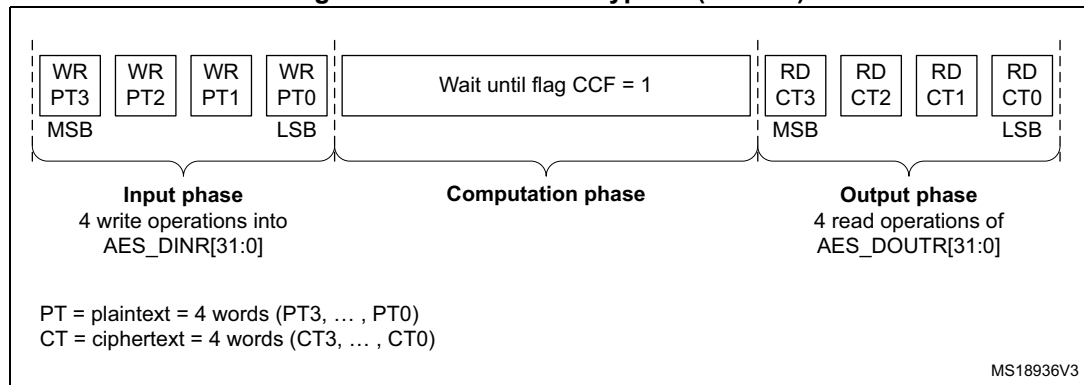
For more information on data swapping, refer to [Section 20.4.13: AES data registers and data swapping](#).

### ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in [Section 20.4.4](#)):

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select the Mode 1 by setting to 00 the MODE[1:0] bitfield of the AES\_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES\_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield.
3. Select 128- or 256-bit key length through the KEYSIZE bit of the AES\_CR register.
4. Write the AES\_KEYRx registers (128 or 256 bits) with encryption key. Fill the AES\_IVRx registers with the initialization vector data if CBC mode has been selected.
5. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
6. Write the AES\_DINR register four times to input the plaintext (MSB first), as shown in [Figure 87](#).
7. Wait until the CCF flag is set in the AES\_SR register.
8. Read the AES\_DOUTR register four times to get the ciphertext (MSB first) as shown in [Figure 87](#). Then clear the CCF flag by setting the CCFC bit of the AES\_CR register.
9. Repeat steps 6-7-8 to process all the blocks with the same encryption key.

**Figure 87. ECB/CBC encryption (Mode 1)**



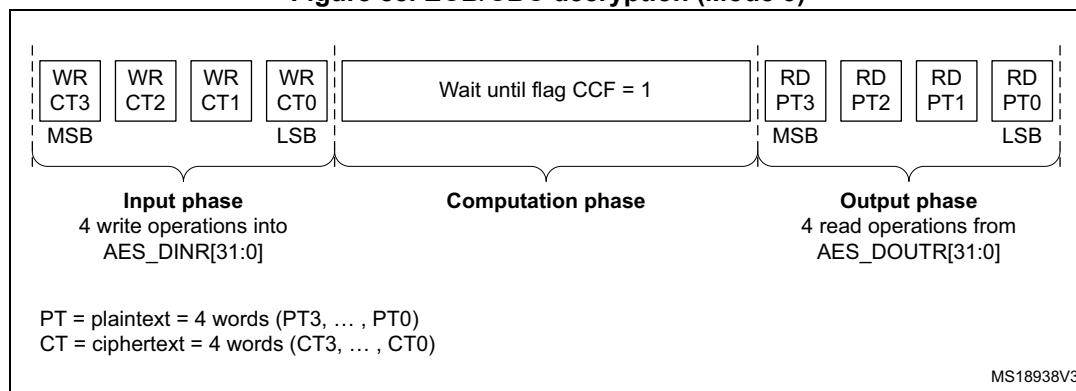
### ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in [Section 20.4.4](#)).

1. Follow the steps described in [Section 20.4.5: AES decryption round key preparation](#), in order to prepare the decryption key in AES core.
2. Select the Mode 3 by setting to 10 the MODE[1:0] bitfield of the AES\_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES\_CR

- register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield. KEYSIZE bitfield must be kept as-is.
- Write the AES\_IVRx registers with the initialization vector (required in CBC mode only).
  - Enable AES by setting the EN bit of the AES\_CR register.
  - Write the AES\_DINR register four times to input the cipher text (MSB first), as shown in [Figure 88](#).
  - Wait until the CCF flag is set in the AES\_SR register.
  - Read the AES\_DOUTR register four times to get the plain text (MSB first), as shown in [Figure 88](#). Then clear the CCF flag by setting the CCFC bit of the AES\_CR register.
  - Repeat steps 5-6-7 to process all the blocks encrypted with the same key.

**Figure 88. ECB/CBC decryption (Mode 3)**



### Suspend/resume operations in ECB/CBC modes

To suspend the processing of a message, proceed as follows:

- If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES\_CR register.
- If DMA is not used, read four times the AES\_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the AES\_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES\_CR register.
- If DMA is not used, poll the CCF flag of the AES\_SR register until it becomes 1 (computation completed).
- Clear the CCF flag by setting the CCFC bit of the AES\_CR register.
- Save initialization vector registers (only required in CBC mode as AES\_IVRx registers are altered during the data processing).
- Disable the AES peripheral by clearing the bit EN of the AES\_CR register.
- Save the AES\_CR register and clear the key registers if they are not needed, to process the higher priority message.
- If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

**To resume the processing of a message**, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
3. Restore AES\_CR register (with correct KEYSIZE) then restore AES\_KEYRx registers.
4. Prepare the decryption key as described in [Section 20.4.5: AES decryption round key preparation](#) (only required for ECB or CBC decryption).
5. Restore AES\_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
7. If DMA is used, enable AES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the AES\_CR register.

#### **Alternative single ECB/CBC decryption using Mode 4**

The sequence of events to perform a single round of ECB/CBC decryption using Mode 4 is:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select the Mode 4 by setting to 11 the MODE[1:0] bitfield of the AES\_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the AES\_CR register to 0x0 or 0x1, respectively.
3. Select key length of 128 or 256 bits via KEYSIZE bitfield of the AES\_CR register.
4. Write the AES\_KEYRx registers with the encryption key. Write the AES\_IVRx registers if the CBC mode is selected.
5. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
6. Write the AES\_DINR register four times to input the cipher text (MSB first).
7. Wait until the CCF flag is set in the AES\_SR register.
8. Read the AES\_DOUTR register four times to get the plain text (MSB first). Then clear the CCF flag by setting the CCFC bit of the AES\_CR register.

*Note:* When mode 4 is selected mode 3 cannot be used.

*In mode 4, the AES\_KEYRx registers contain the encryption key during all phases of the processing. No derivation key is stored in these registers. It is stored internally in AES.*

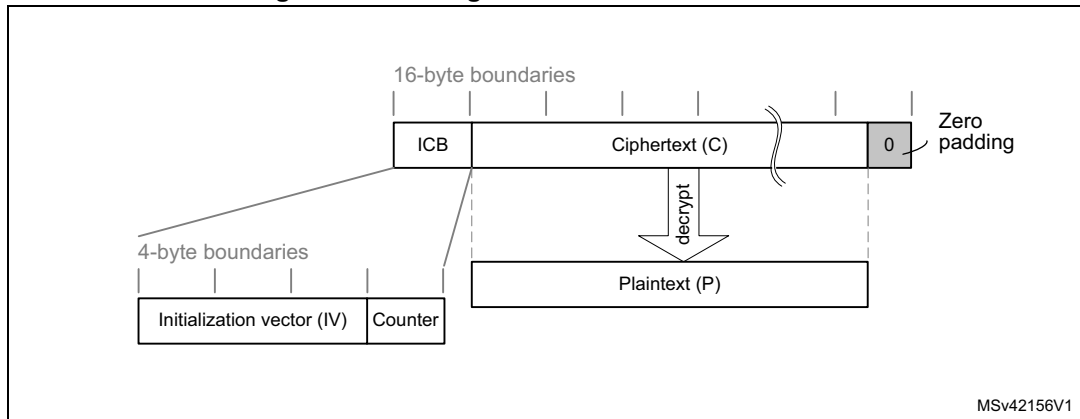
### **20.4.9 AES counter (CTR) mode**

#### **Overview**

The counter mode (CTR) uses AES as a key-stream generator. The generated keys are then XOR-ed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*. A typical message construction in CTR mode is given in [Figure 89](#).

Figure 89. Message construction in CTR mode



The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

**CTR encryption and decryption**

Figure 90 and Figure 91 describe the CTR encryption and decryption process, respectively, as implemented in the AES peripheral. The CTR mode is selected by writing 010 to the CHMOD[2:0] bitfield of AES\_CR register.

Figure 90. CTR encryption

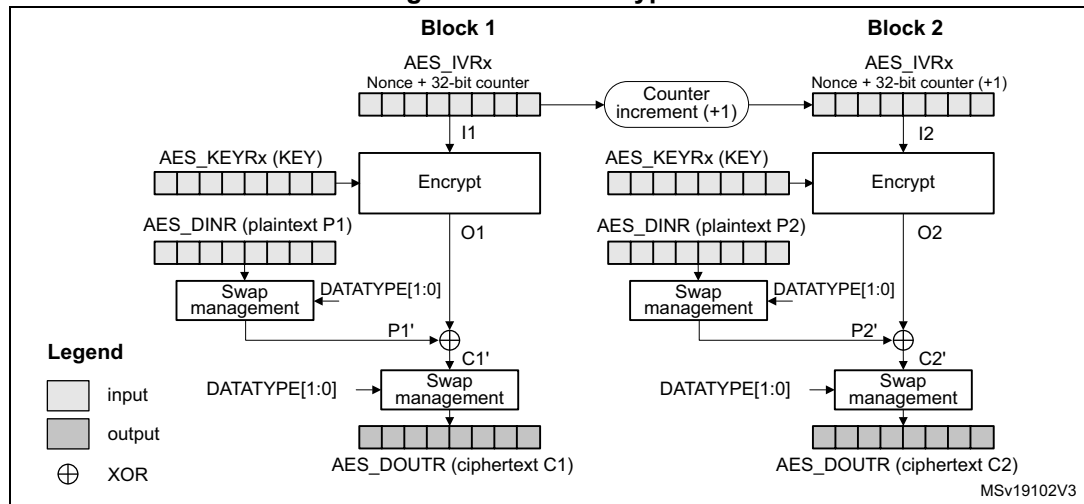
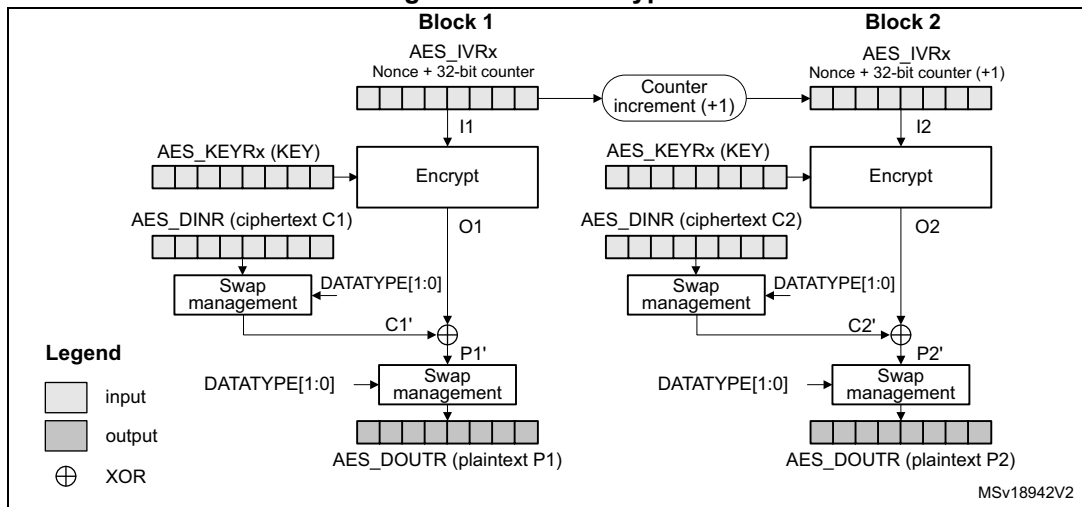


Figure 91. CTR decryption



In CTR mode, the cryptographic core output (also called keystream)  $Ox$  is XOR-ed with relevant input block ( $Px'$  for encryption,  $Cx'$  for decryption), to produce the correct output block ( $Cx'$  for encryption,  $Px'$  for decryption). Initialization vectors in AES must be initialized as shown in [Table 107](#).

Table 107. CTR mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

Unlike in CBC mode that uses the AES\_IVRx registers only once when processing the first data block, in CTR mode AES\_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE[1:0] bitfield setting 01 (key derivation) is forbidden and all the other settings default to encryption mode.



The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bitfield of the AES\_CR register. Set MODE[1:0] bitfield to any value other than 01.
3. Initialize the AES\_KEYRx registers, and load the AES\_IVRx registers as described in [Table 107](#).
4. Set the EN bit of the AES\_CR register, to start encrypting the current counter (EN is automatically reset when the calculation finishes).
5. If it is the last block, pad the data with zeros to have a complete block, if needed.
6. Append data in AES, and read the result. The three possible scenarios are described in [Section 20.4.4: AES procedure to perform a cipher operation](#).
7. Repeat the previous step till the second-last block is processed. For the last block, apply the two previous steps and discard the bits that are not part of the payload (if the size of the significant data in the last input block is less than 16 bytes).

### Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message that was interrupted. Detailed CBC suspend/resume sequence is described in [Section 20.4.8: AES basic chaining modes \(ECB, CBC\)](#).

*Note:* Like for CBC mode, the AES\_IVRx registers must be reloaded during the resume operation.

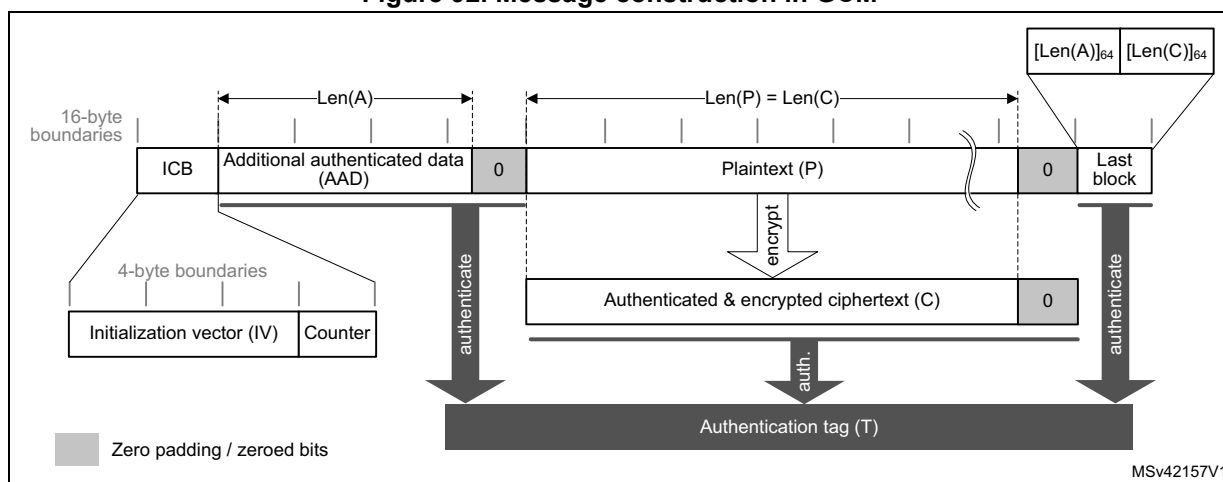
## 20.4.10 AES Galois/counter mode (GCM)

### Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*. A typical message construction in GCM mode is given in [Figure 92](#).

Figure 92. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key. Note that the GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also known as additional authentication data) has a known length  $Len(A)$  that may be a non-multiple of 16 bytes, and must not exceed  $2^{64} - 1$  bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length  $Len(P)$  that may be non-multiple of 16 bytes, and cannot exceed  $2^{32} - 2$  128-bit blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in [Table 108](#).

The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

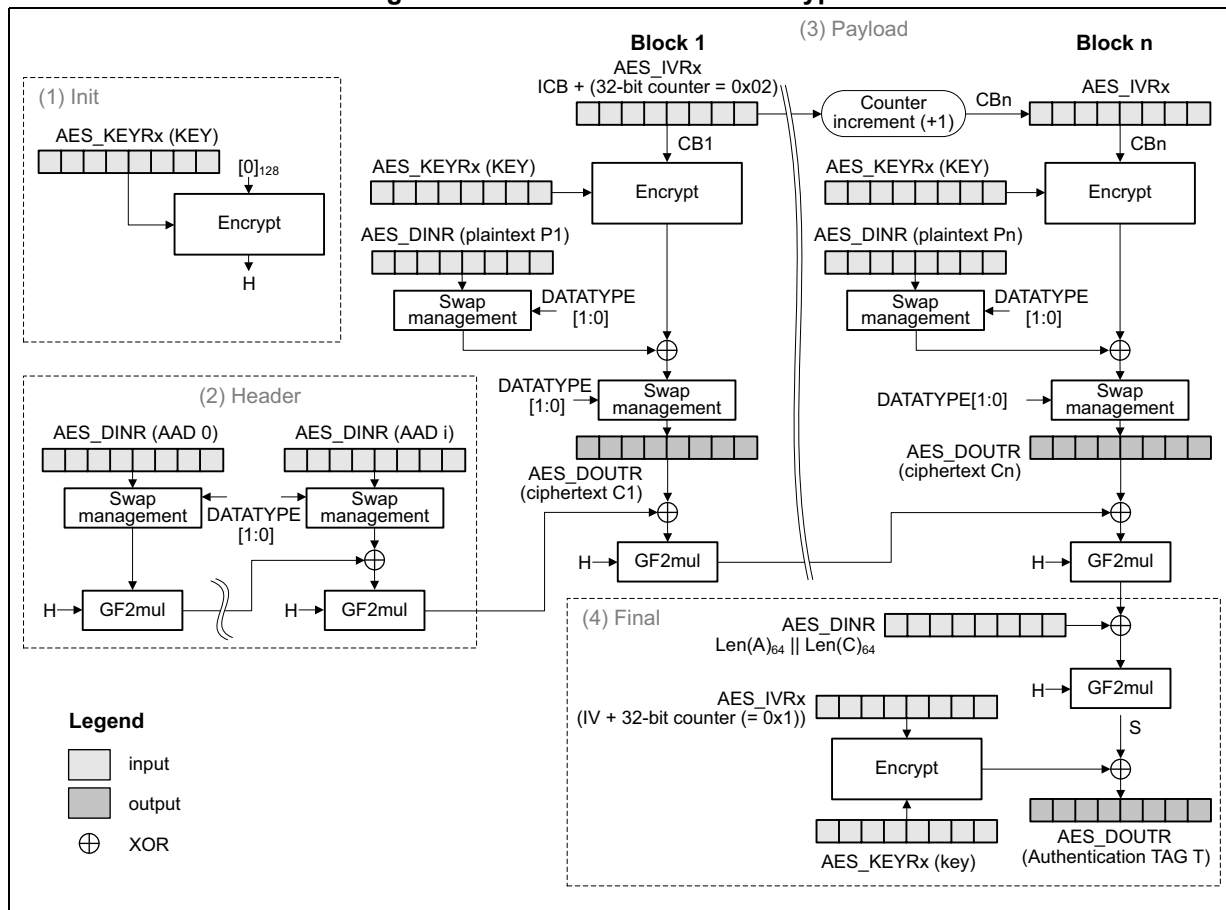
**Table 108. GCM last block definition**

Endianness	Bit[0] ----- Bit[31]	Bit[32]----- Bit[63]	Bit[64] ----- Bit[95]	Bit[96] ----- Bit[127]
Input data	0x0	AAD length[31:0]	0x0	Payload length[31:0]

GCM processing

Figure 93 describes the GCM implementation in the AES peripheral. The GCM is selected by writing 011 to the CHMOD[2:0] bitfield of the AES\_CR register.

Figure 93. GCM authenticated encryption



The mechanism for the confidentiality of the plaintext in GCM mode is similar to that in the Counter mode, with a particular increment function (denoted 32-bit increment) that generates the sequence of input counter blocks.

AES\_IVRx registers keeping the **counter block** of data are used for processing each data block. The AES peripheral automatically increments the Counter[31:0] bitfield. The first counter block (CB1) is derived from the initial counter block ICB by the application software (see Table 109).

Table 109. Initialization of AES\_IVRx registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

Note: In this mode, the settings 01 and 11 of the MODE[1:0] bitfield are forbidden.



The authentication mechanism in GCM mode is based on a hash function called **GF2mul** that performs multiplication by a fixed parameter, called hash subkey (H), within a binary Galois field.

A GCM message is processed through the following phases, further described in next subsections:

- **Init phase:** AES prepares the GCM hash subkey (H).
- **Header phase:** AES processes the additional authenticated data (AAD), with hash computation only.
- **Payload phase:** AES processes the plaintext (P) with hash computation, counter block encryption and data XOR-ing. It operates in a similar way for ciphertext (C).
- **Final phase:** AES generates the authenticated tag (T) using the last block of the message.

### GCM init phase

During this first step, the GCM hash subkey (H) is calculated and saved internally, to be used for processing all the blocks. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select GCM chaining mode, by setting to 011 the CHMOD[2:0] bitfield of the AES\_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES\_CR register.
4. Set the MODE[1:0] bitfield of the AES\_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES\_KEYRx registers with a key, and initialize AES\_IVRx registers with the information as defined in [Table 109](#).
6. Start the calculation of the hash key, by setting to 1 the EN bit of the AES\_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES\_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag of the AES\_SR register, by setting the CCFC bit of the AES\_CR register.

### GCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in [Section 20.4.4: AES procedure to perform a cipher operation](#). No data is read during this phase.
4. Repeat the step 3 until the last additional authenticated data block is processed.

*Note:* The header phase can be skipped if there is no AAD, that is,  $Len(A) = 0$ .

### GCM payload phase

This phase, identical for encryption and decryption, is executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPH[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in [Section 20.4.4: AES procedure to perform a cipher operation on page 473](#), and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, discard the bits that are not part of the payload when the last block size is less than 16 bytes.

*Note:* The payload phase can be skipped if there is no payload data, that is,  $Len(C) = 0$  (see GMAC mode).

### GCM final phase

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPH[1:0] bitfield of the AES\_CR register.
2. Compose the data of the block, by concatenating the AAD bit length and the payload bit length, as shown in [Table 108](#). Write the block into the AES\_DINR register.
3. Wait until the end of computation, indicated by the CCF flag of the AES\_SR transiting to 1.
4. Get the GCM authentication tag, by reading the AES\_DOUTR register four times.
5. Clear the CCF flag of the AES\_SR register, by setting the CCFC bit of the AES\_CR register.
6. Disable the AES peripheral, by clearing the bit EN of the AES\_CR register. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

*Note:* In the final phase, data is written to AES\_DINR normally (no swapping), while swapping is applied to tag data read from AES\_DOUTR.

*When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.*

## Suspend/resume operations in GCM mode

**To suspend the processing of a message**, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES\_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES\_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES\_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the AES\_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES\_CR register.
3. Clear the CCF flag of the AES\_SR register, by setting the CCFC bit of the AES\_CR register.
4. Save the AES\_SUSPxR registers in the memory, where x is from 0 to 7.
5. In the payload phase, save the AES\_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing the EN bit of the AES\_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES\_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

**To resume the processing of a message**, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES\_SUSPxR registers, where x is from 0 to 7.
4. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES\_IVRx registers. In the header phase, write initial setting values back into the AES\_IVRx registers.
5. Restore the initial setting values in the AES\_CR and AES\_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES\_CR register.

If DMA is used, enable AES DMA requests by setting the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES\_CR register.

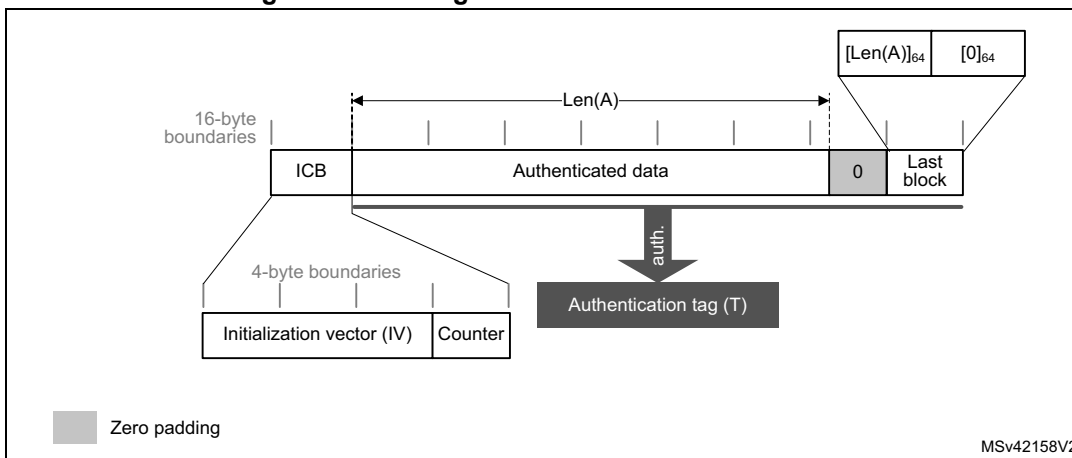
### 20.4.11 AES Galois message authentication code (GMAC)

#### Overview

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

A typical message construction for GMAC is given in [Figure 94](#).

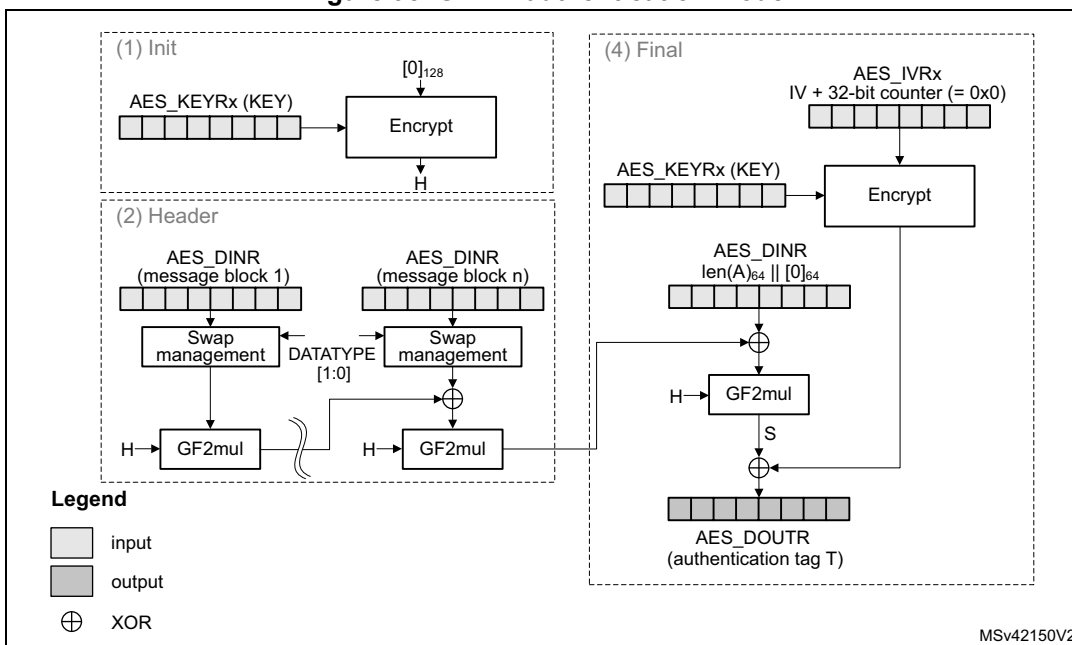
**Figure 94. Message construction in GMAC mode**



**AES GMAC processing**

[Figure 95](#) describes the GMAC mode implementation in the AES peripheral. This mode is selected by writing 011 to the CHMOD[2:0] bitfield of the AES\_CR register.

**Figure 95. GMAC authentication mode**



The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

### Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

## 20.4.12 AES counter with CBC-MAC (CCM)

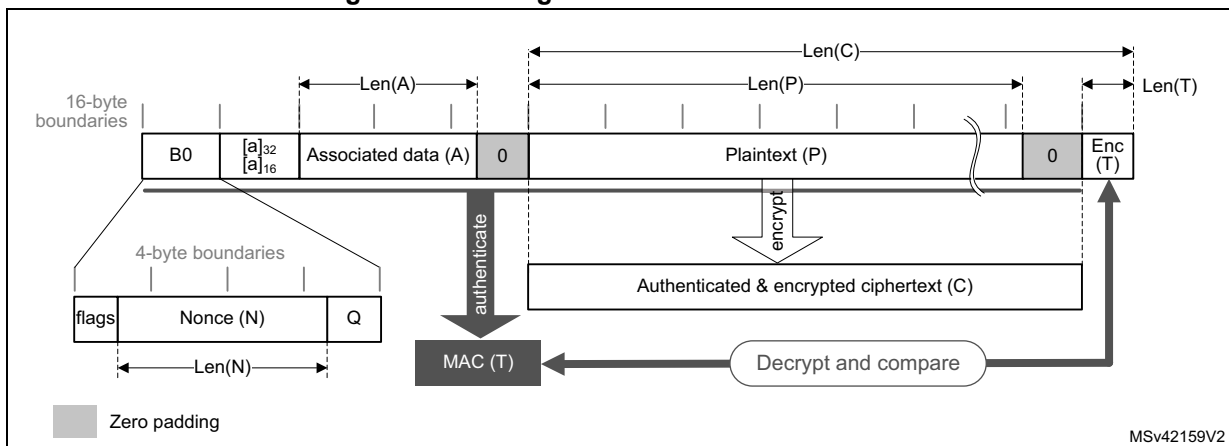
### Overview

The AES **counter with cipher block chaining-message authentication code (CCM)** algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

*Note:* NIST does not approve this CBC-MAC as an authentication mode outside the context of the CCM specification.

CCM chaining is specified in NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*. A typical message construction for CCM is given in [Figure 96](#).

**Figure 96. Message construction in CCM mode**



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q**: a bit string representation of the octet length of P (Len(P))
  - **Nonce (N)**: a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags**: most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2<sup>8Q-4</sup> bytes). The counter blocks range associated to **Q** is equal to 2<sup>8Q-4</sup>, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the Associated Data (A). This part of the message is only authenticated, not encrypted. This section has a



known length  $\text{Len}(A)$  that can be a non-multiple of 16 bytes (see [Figure 96](#)). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes ( $a$ ) must be encoded as follows:

- If  $0 < a < 2^{16} - 2^8$ , then it is encoded as  $[a]_{16}$ , that is, on two bytes.
  - If  $2^{16} - 2^8 < a < 2^{32}$ , then it is encoded as  $0\text{xff} \parallel 0\text{xfe} \parallel [a]_{32}$ , that is, on six bytes.
  - If  $2^{32} < a < 2^{64}$ , then it is encoded as  $0\text{xff} \parallel 0\text{xff} \parallel [a]_{64}$ , that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length  $\text{Len}(P)$ . This length can be a non-multiple of 16 bytes (see [Figure 96](#)).
  - **Encrypted MAC (T)** of length  $\text{Len}(T)$  appended to the ciphertext C of overall length  $\text{Len}(C)$ .

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

*Note:* CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

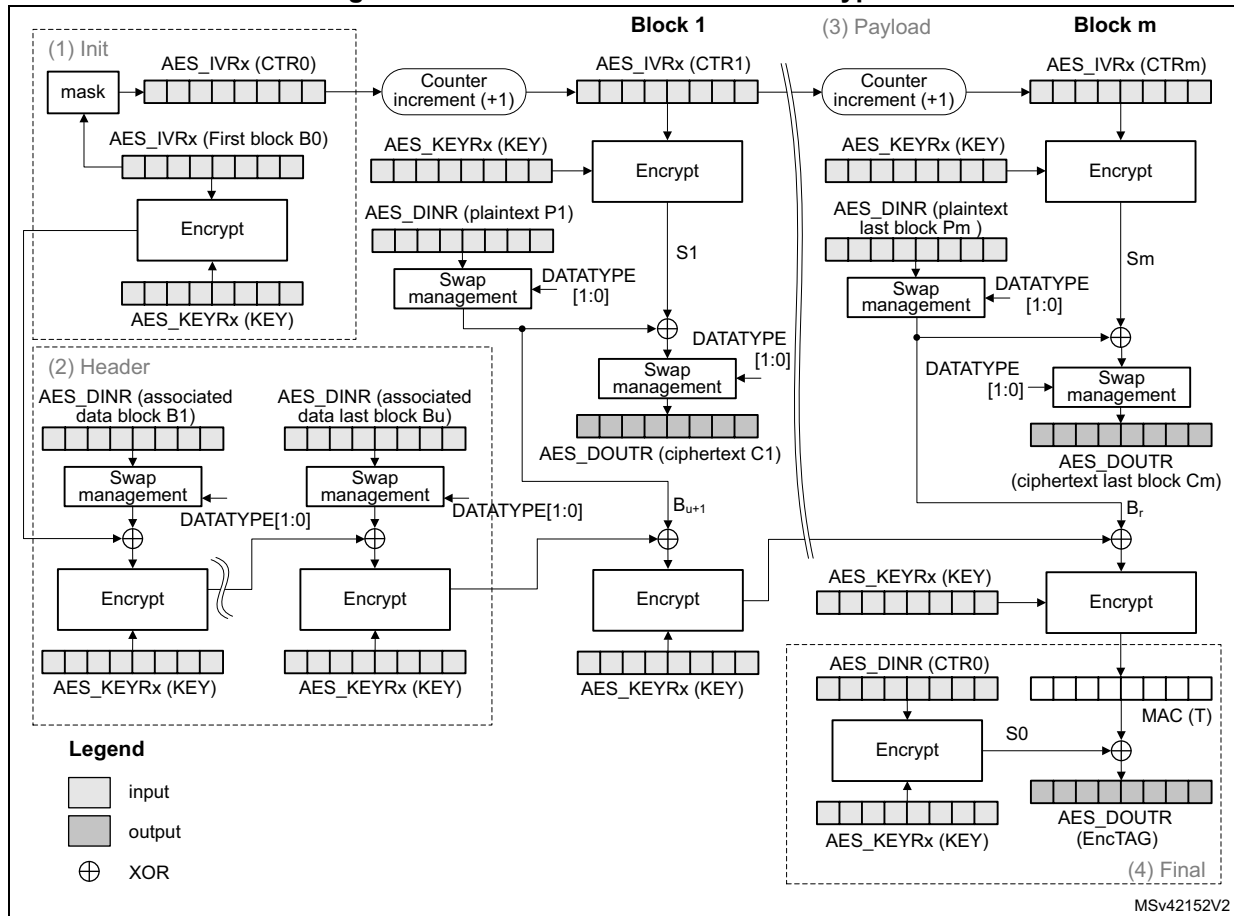
```
N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)
A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)
P: 20212223 (Len(P) = 32 bits or 4 bytes)
T: 6084341B (Len(T) = 32 bits or t = 4)
B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000
CTR0: 0710111213 141516 00000000 00000000
CTR1: 0710111213 141516 00000000 00000001
```

Generation of formatted input data blocks  $B_x$  (especially B0 and B1) must be managed by the application.

CCM processing

Figure 97 describes the CCM implementation within the AES peripheral (encryption example). This mode is selected by writing 100 into the CHMOD[2:0] bitfield of the AES\_CR register.

Figure 97. CCM mode authenticated encryption



The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted plaintext data to generate a MAC, with a known length. Counter mode encryption that requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting ciphertext C is the output of the generation-encryption process on plaintext P.

AES\_IVRx registers are used for processing each data block, AES automatically incrementing the CTR counter with a bit length defined by the first block B0. Table 110 shows how the application must load the B0 data.

Note: The AES peripheral in CCM mode supports counters up to 64 bits, as specified by NIST.

Table 110. Initialization of AES\_IVRx registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96]	B0[95:64]	B0[63:32]	B0[31:0]

*Note:* In this mode, the settings 01 and 11 of the MODE[1:0] bitfield are forbidden.

A CCM message is processed through the following phases, further described in next subsections:

- **Init phase:** AES processes the first block and prepares the first counter block.
- **Header phase:** AES processes associated data (A), with tag computation only.
- **Payload phase:** IP processes plaintext (P), with tag computation, counter block encryption, and data XOR-ing. It works in a similar way for ciphertext (C).
- **Final phase:** AES generates the message authentication code (MAC).

#### CCM Init phase

In this phase, the first block B0 of the CCM message is written into the AES\_IVRx register. The AES\_DOUTR register does not contain any output data. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
2. Select CCM chaining mode, by setting to 100 the CHMOD[2:0] bitfield of the AES\_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES\_CR register.
4. Set the MODE[1:0] bitfield of the AES\_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES\_KEYRx registers with a key, and initialize AES\_IVRx registers with B0 data as described in [Table 110](#).
6. Start the calculation of the counter, by setting to 1 the EN bit of the AES\_CR register (EN is automatically reset when the calculation finishes).
7. Wait until the end of computation, indicated by the CCF flag of the AES\_SR transiting to 1. Alternatively, use the corresponding interrupt.
8. Clear the CCF flag in the AES\_SR register, by setting to 1 the CCFC bit of the AES\_CR register.

#### CCM header phase

This phase coming after the GCM Init phase must be completed before the payload phase. During this phase, the AES\_DOUTR register does not contain any output data.

The sequence to execute, identical for encryption and decryption, is:

1. Indicate the header phase, by setting to 01 the GCMPH[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last block and the AAD size in the block is inferior to 128 bits, pad the remainder of the block with zeros. Then append the data block into AES in one of ways described in [Section 20.4.4: AES procedure to perform a cipher operation](#). No data is read during this phase.
4. Repeat the step 3 until the last additional authenticated data block is processed.

*Note:* The header phase can be skipped if there is no associated data, that is,  $Len(A) = 0$ .  
The first block of the associated data (B1) must be formatted by software, with the associated data length.

**CCM payload phase (encryption or decryption)**

This phase, identical for encryption and decryption, is executed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the payload phase, by setting to 10 the GCMPPH[1:0] bitfield of the AES\_CR register. Do not modify the MODE[1:0] bitfield as set in the Init phase.
2. If the header phase was skipped, enable the AES peripheral by setting the EN bit of the AES\_CR register.
3. If it is the last data block to encrypt and the plaintext size in the block is inferior to 128 bits, pad the remainder of the block with zeros.
4. Append the data block into AES in one of ways described in [Section 20.4.4: AES procedure to perform a cipher operation on page 473](#), and read the result.
5. Repeat the previous step till the second-last plaintext block is encrypted or till the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), apply the two previous steps. For the last block, discard the data that is not part of the payload when the last block size is less than 16 bytes.

*Note:* The payload phase can be skipped if there is no payload data, that is,  $Len(P) = 0$  or  $Len(C) = Len(T)$ .

*Remove  $LSB_{Len(T)}(C)$  encrypted tag information when decrypting ciphertext C.*

**CCM final phase**

In this last phase, the AES peripheral generates the GCM authentication tag and stores it in the AES\_DOUTR register. The sequence to execute is:

1. Indicate the final phase, by setting to 11 the GCMPPH[1:0] bitfield of the AES\_CR register.
2. Wait until the end-of-computation flag CCF of the AES\_SR register is set.
3. Read four times the AES\_DOUTR register: the output corresponds to the CCM authentication tag.
4. Clear the CCF flag of the AES\_SR register by setting the CCFC bit of the AES\_CR register.
5. Disable the AES peripheral, by clearing the EN bit of the AES\_CR register.
6. For authenticated decryption, compare the generated encrypted tag with the encrypted tag padded in the ciphertext.

*Note:* In this final phase, swapping is applied to tag data read from AES\_DOUTR register.

*When transiting from the header phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.*

*Application must mask the authentication tag output with tag length to obtain a valid tag.*

**Suspend/resume operations in CCM mode**

**To suspend the processing of a message** in header or payload phase, proceed as follows:

1. If DMA is used, stop the AES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the AES\_CR register. If DMA is not used, make sure that the current computation is completed, which is indicated by the CCF flag of the AES\_SR register set to 1.
2. In the payload phase, if DMA is not used, read four times the AES\_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag is set in the

AES\_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the AES\_CR register.

3. Clear the CCF flag of the AES\_SR register, by setting to 1 the CCFC bit of the AES\_CR register.
4. Save the AES\_SUSPxR registers (where x is from 0 to 7) in the memory.
5. Save the AES\_IVRx registers as, during the data processing, they changed from their initial values.
6. Disable the AES peripheral, by clearing the EN bit of the AES\_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES\_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

**To resume the processing of a message**, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES\_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES\_SUSPxR registers (where x is from 0 to 7).
4. Write the initialization vector register values, previously saved in the memory, back into their corresponding AES\_IVRx registers.
5. Restore the initial setting values in the AES\_CR and AES\_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES\_CR register.
7. If DMA is used, enable AES DMA requests by setting to 1 the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES\_CR register.

### 20.4.13 AES data registers and data swapping

#### Data input and output

A 128-bit data block is entered into the AES peripheral with four successive 32-bit word writes into the AES\_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 128-bit data block is retrieved from the AES peripheral with four successive 32-bit word reads from the AES\_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The 32-bit data word for AES\_DINR register or from AES\_DOUTR register is organized in big endian order, that is:

- the most significant byte of a word to write into AES\_DINR must be put on the lowest address out of the four adjacent memory locations keeping the word to write, or
- the most significant byte of a word read from AES\_DOUTR goes to the lowest address out of the four adjacent memory locations receiving the word

For using DMA for input data block write into AES, the four words of the input block must be stored in the memory consecutively and in big-endian order, that is, the most significant word on the lowest address. See [Section 20.4.16: AES DMA interface](#).

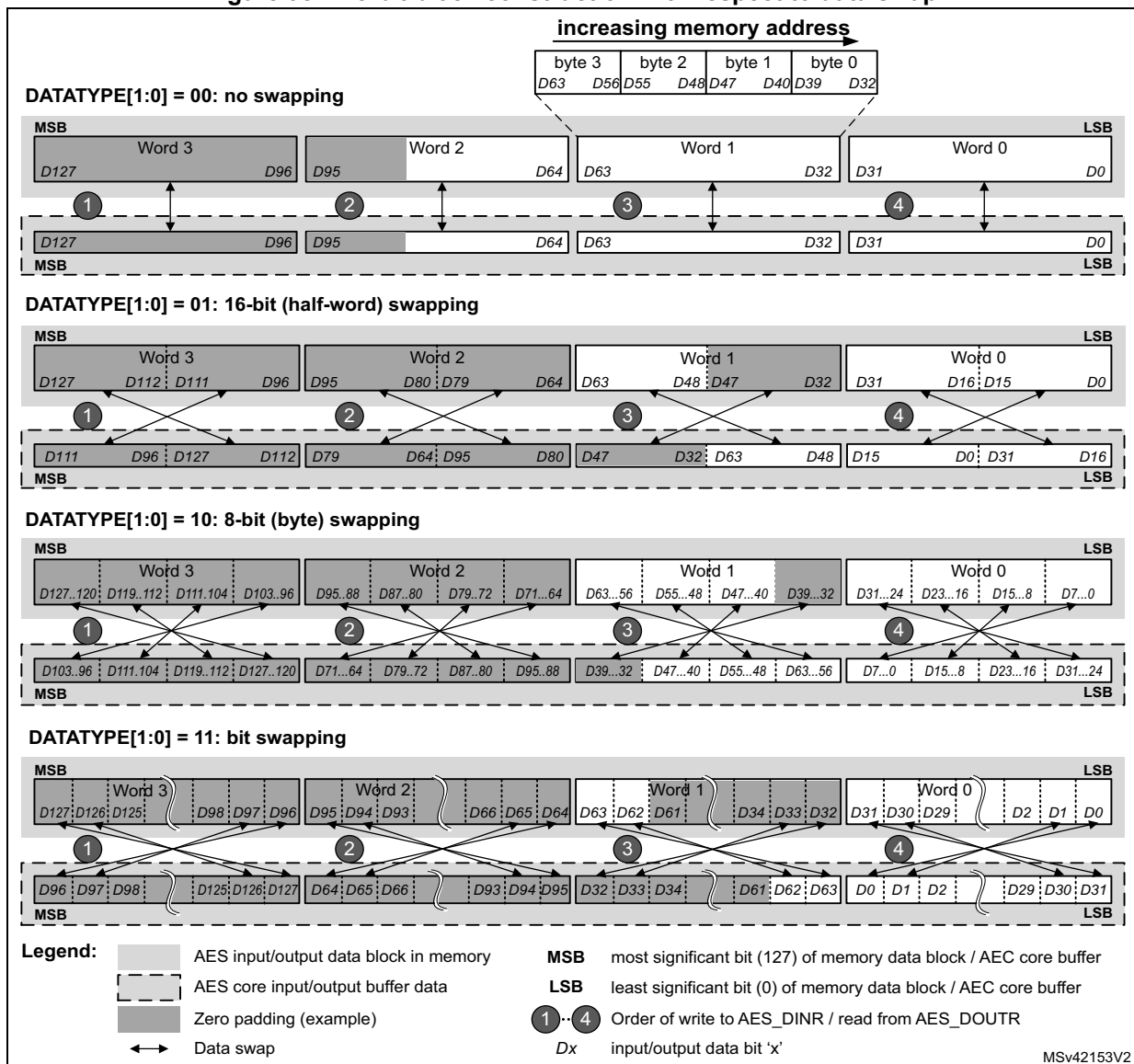
### Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES\_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES\_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0] bitfield of the AES\_CR register. The selection applies both to the input and the output of the AES core.

For different data swap types, *Figure 98* shows the construction of AES processing core input buffer data P127 to P0, from the input data entered through the AES\_DINR register, or the construction of the output data available through the AES\_DOUTR register, from the AES processing core output buffer data P127 to P0.

**Figure 98. 128-bit block construction with respect to data swap**



*Note:* The data in AES key registers (AES\_KEYRx) and initialization registers (AES\_IVRx) are not sensitive to the swap mode selection.

**Data padding**

Figure 98 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 48 message bits, with DATATYPE[1:0] = 01
- 56 message bits, with DATATYPE[1:0] = 10
- 34 message bits, with DATATYPE[1:0] = 11

**20.4.14 AES key registers**

The AES\_KEYRx write-only registers store the encryption or decryption key bitfield KEY[127:0] or KEY[255:0]. The data to write to each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address (reads are not allowed for security reason).

The key is spread over eight registers as shown in Table 111.

**Table 111. Key endianness in AES\_KEYRx registers (128- or 256-bit key length)**

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key for encryption or decryption may be written into these registers when the AES peripheral is disabled, by clearing the EN bit of the AES\_CR register.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the AES\_CR register.

**20.4.15 AES initialization vector registers**

The four AES\_IVRx registers keep the initialization vector input bitfield IVI[127:0]. The data to write to or to read from each register is organized in the memory in little-endian order, that is, with most significant byte on the highest address. The registers are also ordered from lowest address (AES\_IVR0) to highest address (AES\_IVR3).

The signification of data in the bitfield depends on the chaining mode selected. When used, the bitfield is updated upon each computation cycle of the AES core.

Write operations to the AES\_IVRx registers when the AES peripheral is enabled have no effect to the register contents. For modifying the contents of the AES\_IVRx registers, the EN bit of the AES\_CR register must first be cleared.

Reading the AES\_IVRx registers returns the latest counter value (useful for managing suspend mode).

The AES\_IVRx registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield of the AES\_CR register.



### 20.4.16 AES DMA interface

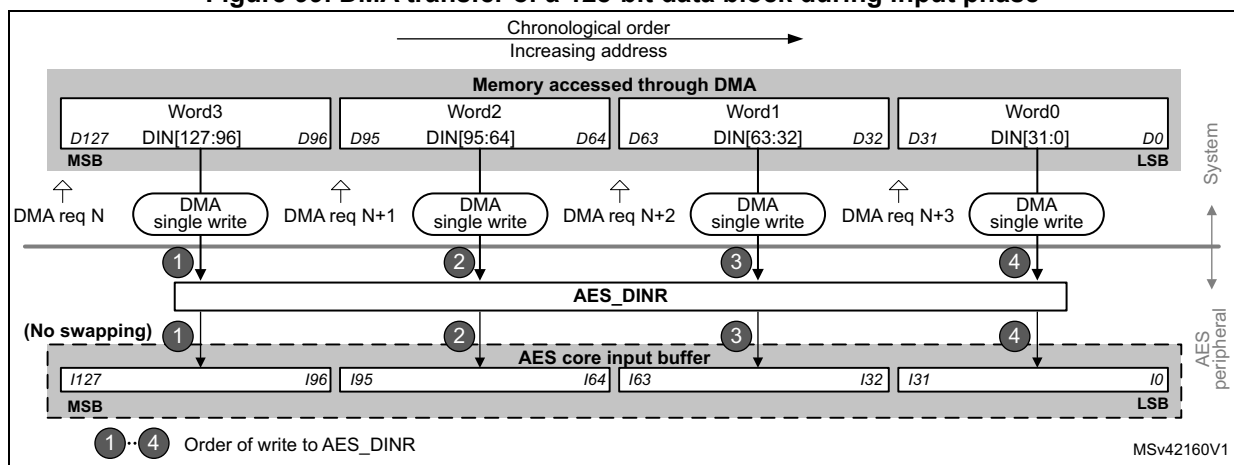
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES\_CR register.

#### Data input using DMA

Setting the DMAINEN bit of the AES\_CR register enables DMA writing into AES. The AES peripheral then initiates a DMA request during the input phase each time it requires to write a 128-bit block (quadruple word) to the AES\_DINR register, as shown in [Figure 99](#).

*Note:* According to the algorithm and the mode selected, special padding / ciphertext stealing might be required. For example, in case of AES GCM encryption or AES CCM decryption, a DMA transfer must not include the last block. For details, refer to [Section 20.4.4: AES procedure to perform a cipher operation](#).

**Figure 99. DMA transfer of a 128-bit data block during input phase**



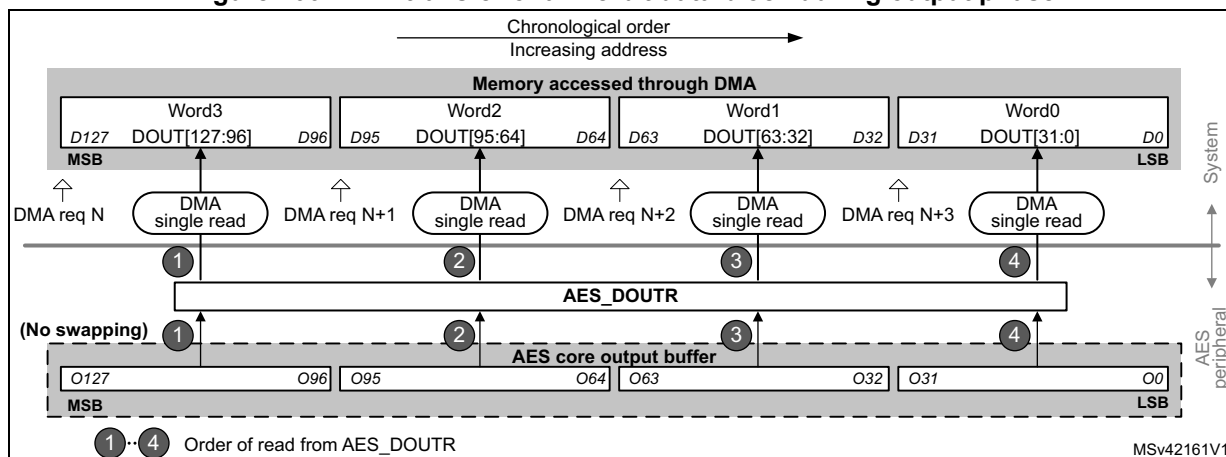
#### Data output using DMA

Setting the DMAOUTEN bit of the AES\_CR register enables DMA reading from AES. The AES peripheral then initiates a DMA request during the Output phase each time it requires to read a 128-bit block (quadruple word) to the AES\_DINR register, as shown in [Figure 100](#).

*Note:* According to the message size, extra bytes might need to be discarded by application in the last block.



Figure 100. DMA transfer of a 128-bit data block during output phase



### DMA operation in different operating modes

DMA operations are usable when Mode 1 (encryption) or Mode 3 (decryption) are selected via the MODE[1:0] bitfield of the register AES\_CR. As in Mode 2 (key derivation) the AES\_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the AES\_CR register have no effect in that mode.

DMA single requests are generated by AES until it is disabled. So, after the data output phase at the end of processing of a 128-bit data block, AES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between AES and memory is managed by DMA, the CCF flag has no use because the reading of the AES\_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to data transferring managed by software. See [Section 20.4.4: AES procedure to perform a cipher operation](#), subsection [Data append](#), for details.

## 20.4.17 AES error management

AES configuration can be changed at any moment by clearing the EN bit of the AES\_CR register.

### Read error flag (RDERR)

Unexpected read attempt of the AES\_DOUTR register sets the RDERR flag of the AES\_SR register, and returns zero.

RDERR is triggered during the computation phase or during the input phase.

*Note:* AES is not disabled upon a RDERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES\_CR register is set. For more details, refer to [Section 20.5: AES interrupts](#).

The RDERR flag is cleared by setting the ERRIE bit of the AES\_CR register.

### Write error flag (WDERR)

Unexpected write attempt of the AES\_DINR register sets the WRERR flag of the AES\_SR register, and has no effect on the AES\_DINR register. The WRERR is triggered during the computation phase or during the output phase.

*Note:* AES is not disabled after a WRERR error detection and continues processing.

An interrupt is generated if the ERRIE bit of the AES\_CR register is set. For more details, refer to [Section 20.5: AES interrupts](#).

The WRERR flag is cleared by setting the ERRC bit of the AES\_CR register.

## 20.5 AES interrupts

Individual maskable interrupt sources generated by the AES peripheral signal the following events:

- computation completed
- read error
- write error

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Arm® Cortex® interrupt controller. Each can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the AES\_CR register, and cleared by setting the corresponding bit of the AES\_CR register.

The status of each can be read from the AES\_SR register.

[Table 112](#) gives a summary of the interrupt sources, their event flags and enable bits.

**Table 112. AES interrupt requests**

Interrupt acronym	AES interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCFC <sup>(1)</sup>
	read error flag	RDERR	ERRIE	set ERRC <sup>(1)</sup>
	write error flag	WRERR		

1. Bit of the AES\_CR register.

## 20.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

**Table 113. Processing latency for ECB, CBC and CTR**

Key size	Mode of operation	Algorithm	Clock cycles
128-bit	Mode 1: Encryption	ECB, CBC, CTR	51
	Mode 2: Key derivation	-	59
	Mode 3: Decryption	ECB, CBC, CTR	51
	Mode 4: Key derivation then decryption	ECB, CBC	106

**Table 113. Processing latency for ECB, CBC and CTR (continued)**

Key size	Mode of operation	Algorithm	Clock cycles
256-bit	Mode 1: Encryption	ECB, CBC, CTR	75
	Mode 2: Key derivation	-	82
	Mode 3: Decryption	ECB, CBC, CTR	75
	Mode 4: Key derivation then decryption	ECB, CBC	145

**Table 114. Processing latency for GCM and CCM (in clock cycles)**

Key size	Mode of operation	Algorithm	Init Phase	Header phase <sup>(1)</sup>	Payload phase <sup>(1)</sup>	Tag phase <sup>(1)</sup>
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

## 20.7 AES registers

### 20.7.1 AES control register (AES\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
								rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:20 **NPBLB[3:0]**: Number of padding bytes in last block

The bitfield sets the number of padding bytes in last block of payload:

0000: All bytes are valid (no padding)

0001: Padding for one least-significant byte of last block

...

1111: Padding for 15 least-significant bytes of last block

Bit 19 Reserved, must be kept at reset value.

- Bit 18 **KEYSIZE**: Key size selection  
This bitfield defines the length of the key used in the AES cryptographic core, in bits:  
0: 128  
1: 256  
Attempts to write the bit are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access.
- Bit 17 Reserved, must be kept at reset value.
- Bit 15 Reserved, must be kept at reset value.
- Bits 14:13 **GCMPTH[1:0]**: GCM or CCM phase selection  
This bitfield selects the phase of GCM, GMAC or CCM algorithm:  
00: Init phase  
01: Header phase  
10: Payload phase  
11: Final phase  
The bitfield has no effect if other than GCM, GMAC or CCM algorithms are selected (through the ALGOMODE bitfield).
- Bit 12 **DMAOUTEN**: DMA output enable  
This bit enables/disables data transferring with DMA, in the output phase:  
0: Disable  
1: Enable  
When the bit is set, DMA requests are automatically generated by AES during the output data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).  
Use of DMA with Mode 4 (single decryption) is not recommended.
- Bit 11 **DMAINEN**: DMA input enable  
This bit enables/disables data transferring with DMA, in the input phase:  
0: Disable  
1: Enable  
When the bit is set, DMA requests are automatically generated by AES during the input data phase. This feature is only effective when Mode 1 or Mode 3 is selected through the MODE[1:0] bitfield. It is not effective for Mode 2 (key derivation).  
Use of DMA with Mode 4 (single decryption) is not recommended.
- Bit 10 **ERRIE**: Error interrupt enable  
This bit enables or disables (masks) the AES interrupt generation when RDERR and/or WRERR is set:  
0: Disable (mask)  
1: Enable
- Bit 9 **CCFIE**: CCF interrupt enable  
This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set:  
0: Disable (mask)  
1: Enable
- Bit 8 **ERRC**: Error flag clear  
Upon written to 1, this bit clears the RDERR and WRERR error flags in the AES\_SR register:  
0: No effect  
1: Clear RDERR and WRERR flags  
Reading the flag always returns zero.

**Bit 7 CCFC:** Computation complete flag clear

Upon written to 1, this bit clears the computation complete flag (CCF) in the AES\_SR register:

0: No effect

1: Clear CCF

Reading the flag always returns zero.

**Bits 16, 6:5 CHMOD[2:0]:** Chaining mode selection

This bitfield selects the AES chaining mode:

000: Electronic codebook (ECB)

001: Cipher-block chaining (CBC)

010: Counter mode (CTR)

011: Galois counter mode (GCM) and Galois message authentication code (GMAC)

100: Counter with CBC-MAC (CCM)

others: Reserved

Attempts to write the bitfield are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access.

**Bits 4:3 MODE[1:0]:** AES operating mode

This bitfield selects the AES operating mode:

00: Mode 1: encryption

01: Mode 2: key derivation (or key preparation for ECB/CBC decryption)

10: Mode 3: decryption

11: Mode 4: key derivation then single decryption

Attempts to write the bitfield are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access. Any attempt to selecting Mode 4 while either ECB or CBC chaining mode is not selected, defaults to effective selection of Mode 3. It is not possible to select a Mode 3 following a Mode 4.

**Bits 2:1 DATATYPE[1:0]:** Data type selection

This bitfield defines the format of data written in the AES\_DINR register or read from the AES\_DOUTR register, through selecting the mode of data swapping:

00: None

01: Half-word (16-bit)

10: Byte (8-bit)

11: Bit

For more details, refer to [Section 20.4.13: AES data registers and data swapping](#).

Attempts to write the bitfield are ignored when the EN bit of the AES\_CR register is set before the write access and it is not cleared by that write access.

**Bit 0 EN:** AES enable

This bit enables/disables the AES peripheral:

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the AES peripheral.

This bit is automatically cleared by hardware upon the completion of the key preparation (Mode 2) and upon the completion of GCM/GMAC/CCM initial phase.

### 20.7.2 AES status register (AES\_SR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
												r	r	r	r

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

**Bit 3 BUSY:** Busy

This flag indicates whether AES is idle or busy during GCM payload **encryption** phase:

0: Idle

1: Busy

When the flag indicates “idle”, the current GCM encryption processing may be suspended to process a higher-priority message. In other chaining modes, or in GCM phases other than payload encryption, the flag must be ignored for the suspend process.

**Bit 2 WRERR:** Write error

This flag indicates the detection of an unexpected write operation to the AES\_DINR register (during computation or data output phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES\_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES\_CR register.

The flag setting has no impact on the AES operation. Unexpected write is ignored.

**Bit 1 RDERR:** Read error flag

This flag indicates the detection of an unexpected read operation from the AES\_DOUTR register (during computation or data input phase):

0: Not detected

1: Detected

The flag is set by hardware. It is cleared by software upon setting the ERRC bit of the AES\_CR register.

Upon the flag setting, an interrupt is generated if enabled through the ERRIE bit of the AES\_CR register.

The flag setting has no impact on the AES operation. Unexpected read returns zero.

**Bit 0 CCF:** Computation completed flag

This flag indicates whether the computation is completed:

0: Not completed

1: Completed

The flag is set by hardware upon the completion of the computation. It is cleared by software, upon setting the CCFC bit of the AES\_CR register.

Upon the flag setting, an interrupt is generated if enabled through the CCFIE bit of the AES\_CR register.

The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMA\_EN is 1.

### 20.7.3 AES data input register (AES\_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

Only 32-bit access type is supported.

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

Bits 31:0 **DIN[31:0]**: Input data word

A four-fold sequential write to this bitfield during the input phase results in writing a complete 128-bit block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 128-bit input buffer.

The data signification of the input data block depends on the AES operating mode:

- **Mode 1** (encryption): plaintext
- **Mode 2** (key derivation): the bitfield is not used (AES\_KEYRx registers used for input)
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): ciphertext

The data swap operation is described in [Section 20.4.13: AES data registers and data swapping on page 497](#).

### 20.7.4 AES data output register (AES\_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

Only 32-bit read access type is supported.

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

Bits 31:0 **DOUT[31:0]**: Output data word

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF set), virtually reads a complete 128-bit block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

The data signification of the output data block depends on the AES operating mode:

- **Mode 1** (encryption): ciphertext
- **Mode 2** (key derivation): the bitfield is not used
- **Mode 3** (decryption) and **Mode 4** (key derivation then single decryption): plaintext

The data swap operation is described in [Section 20.4.13: AES data registers and data swapping on page 497](#).

### 20.7.5 AES key register 0 (AES\_KEYR0)

Address offset: 0x10

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]**: Cryptographic key, bits [31:0]

This write-only bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode:

- In **Mode 1** (encryption), **Mode 2** (key derivation) and **Mode 4** (key derivation then single decryption): the value to write into the bitfield is the encryption key.
- In **Mode 3** (decryption): the value to write into the bitfield is the encryption key to be derived before being used for decryption.

*Note:* In mode 4 (key derivation then single decryption) the bitfield always contains the encryption key.

The AES\_KEYRx registers may be written only when KEYSIZE value is correct and when the AES peripheral is disabled (EN bit of the AES\_CR register cleared). Note that, if, the key is directly loaded to AES\_KEYRx registers (hence writes to key register is ignored and KEIF is set).

Refer to [Section 20.4.14: AES key registers on page 499](#) for more details.

### 20.7.6 AES key register 1 (AES\_KEYR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
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[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w



Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 20.7.7 AES key register 2 (AES\_KEYR2)

Address offset: 0x18  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
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[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 20.7.8 AES key register 3 (AES\_KEYR3)

Address offset: 0x1C  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
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[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

### 20.7.9 AES initialization vector register 0 (AES\_IVR0)

Address offset: 0x20  
 Reset value: 0x0000 0000

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

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

Refer to [Section 20.4.15: AES initialization vector registers on page 499](#) for description of the IVI[127:0] bitfield.

The initialization vector is only used in chaining modes other than ECB.

The AES\_IVRx registers may be written only when the AES peripheral is disabled

### 20.7.10 AES initialization vector register 1 (AES\_IVR1)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

### 20.7.11 AES initialization vector register 2 (AES\_IVR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

### 20.7.12 AES initialization vector register 3 (AES\_IVR3)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]  
 Refer to the AES\_IVR0 register for description of the IVI[128:0] bitfield.

**20.7.13 AES key register 4 (AES\_KEYR4)**

Address offset: 0x30  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
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[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

**20.7.14 AES key register 5 (AES\_KEYR5)**

Address offset: 0x34  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
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[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

**20.7.15 AES key register 6 (AES\_KEYR6)**

Address offset: 0x38  
 Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
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[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]  
 Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.



### 20.7.16 AES key register 7 (AES\_KEYR7)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
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[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the AES\_KEYR0 register for description of the KEY[255:0] bitfield.

*Note:* The key registers from 4 to 7 are used only when the key length of 256 bits is selected. They have no effect when the key length of 128 bits is selected (only key registers 0 to 3 are used in that case).

### 20.7.17 AES suspend registers (AES\_SUSPxR)

Address offset: 0x040 + 0x4 \* x, (x = 0 to 7)

Reset value: 0x0000 0000

These registers contain the complete internal register states of the AES processor when the AES processing of the current task is suspended to process a higher-priority task.

Upon suspend, the software reads and saves the AES\_SUSPxR register contents (where x is from 0 to 7) into memory, before using the AES processor for the higher-priority task.

Upon completion, the software restores the saved contents back into the corresponding suspend registers, before resuming the original task.

*Note:* These registers are used only when GCM, GMAC, or CCM chaining mode is selected.

*These registers can be read only when AES is enabled. Reading these registers while AES is disabled returns 0x0000 0000.*

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

Bits 31:0 **SUSP[31:0]**: AES suspend

Upon suspend operation, this bitfield of the corresponding AES\_SUSPxR register takes the value of one of internal AES registers.

20.7.18 AES register map

Table 115. AES 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	AES_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMMPH[1:0]		DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	Res.	CHMOD[1:0]	Res.	MODE[1:0]		Res.	DATATYPE[1:0]		EN
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x004	AES_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.	BUSY	WRERR	RDERR	CCF	
	Reset value																													0	0	0	0		
0x008	AES_DINR	DIN[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00C	AES_DOUTR	DOUT[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x010	AES_KEYR0	KEY[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x014	AES_KEYR1	KEY[63:32]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x018	AES_KEYR2	KEY[95:64]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x01C	AES_KEYR3	KEY[127:96]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x020	AES_IVR0	IVI[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x024	AES_IVR1	IVI[63:32]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x028	AES_IVR2	IVI[95:64]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x02C	AES_IVR3	IVI[127:96]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x030	AES_KEYR4	KEY[159:128]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x034	AES_KEYR5	KEY[191:160]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x038	AES_KEYR6	KEY[223:192]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x03C	AES_KEYR7	KEY[255:224]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x040	AES_SUSP0R	SUSP[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	



**Table 115. AES 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
0x044	AES_SUSP1R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSP2R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSP3R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSP4R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSP5R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSP6R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSP7R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x3FF	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.	Res.

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 21 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.

### 21.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 21.3.26: Timer synchronization](#).

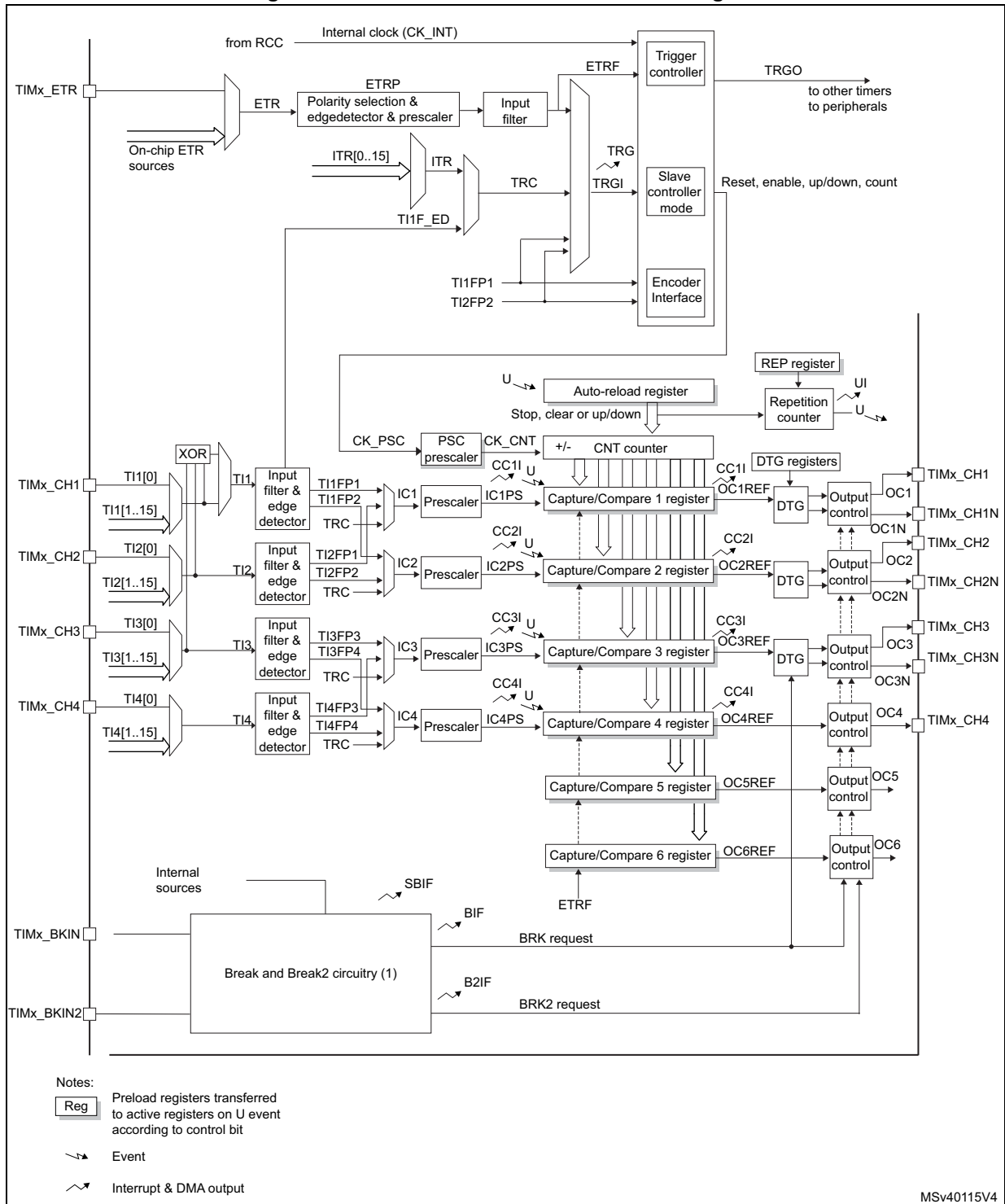
## 21.2 TIM1 main features

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 101. Advanced-control timer block diagram



- The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 5.2.9: Clock security system \(CSS\)](#)
  - A PVD output
  - SRAM parity error signal
  - Cortex<sup>®</sup>-M0+ LOCKUP (Hardfault) output.
  - COMPx output, x=1,2,3 (COMP3 is available on STM32G0B1xx and STM32G0C1xx salestypes only).

## 21.3 TIM1 functional description

### 21.3.1 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 102* and *Figure 103* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 102. Counter timing diagram with prescaler division change from 1 to 2

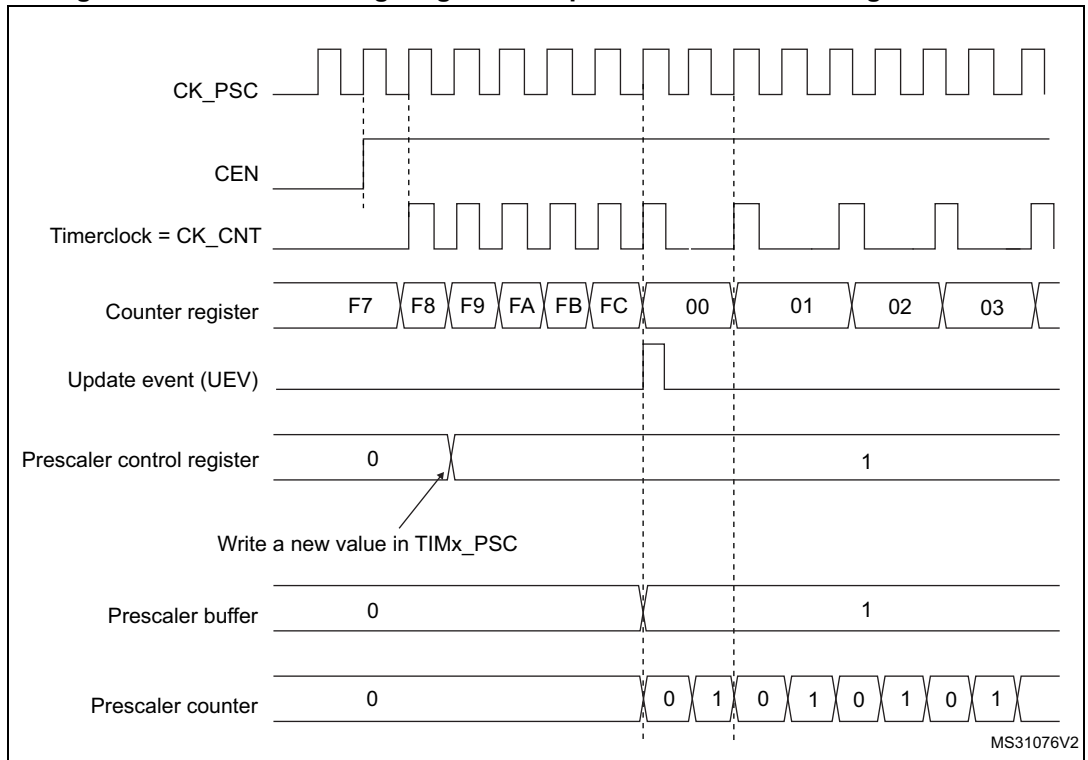
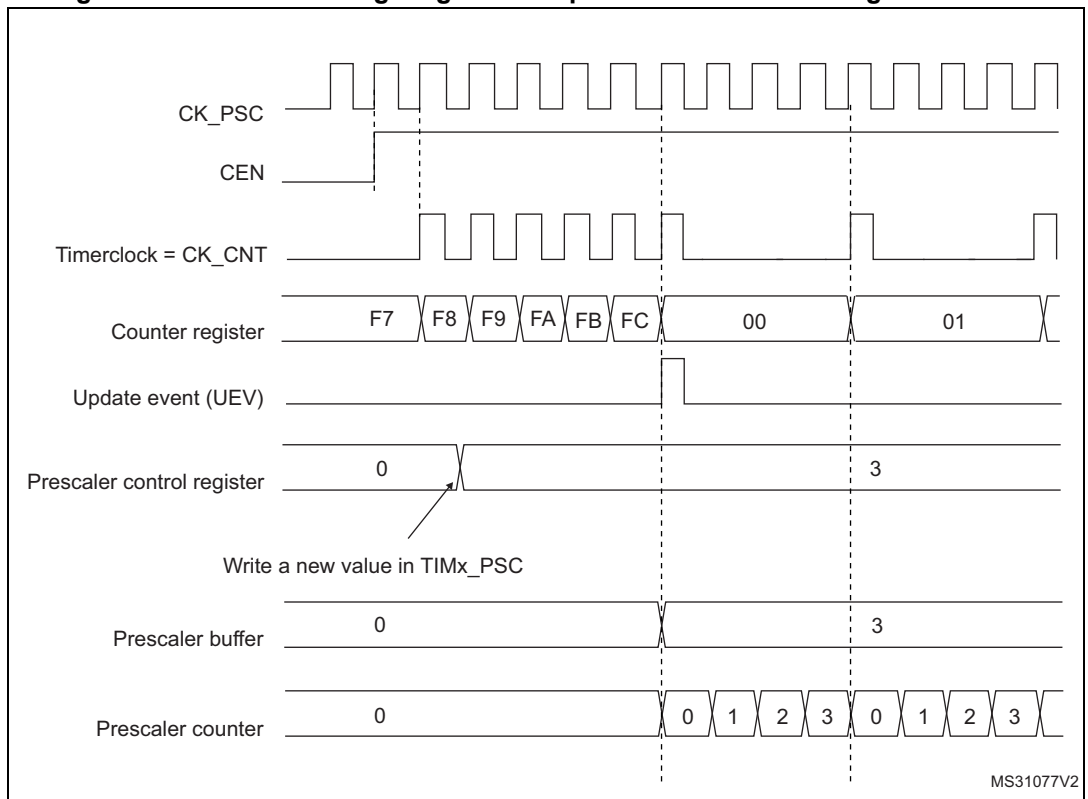


Figure 103. Counter timing diagram with prescaler division change from 1 to 4



## 21.3.2 Counter modes

### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

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.

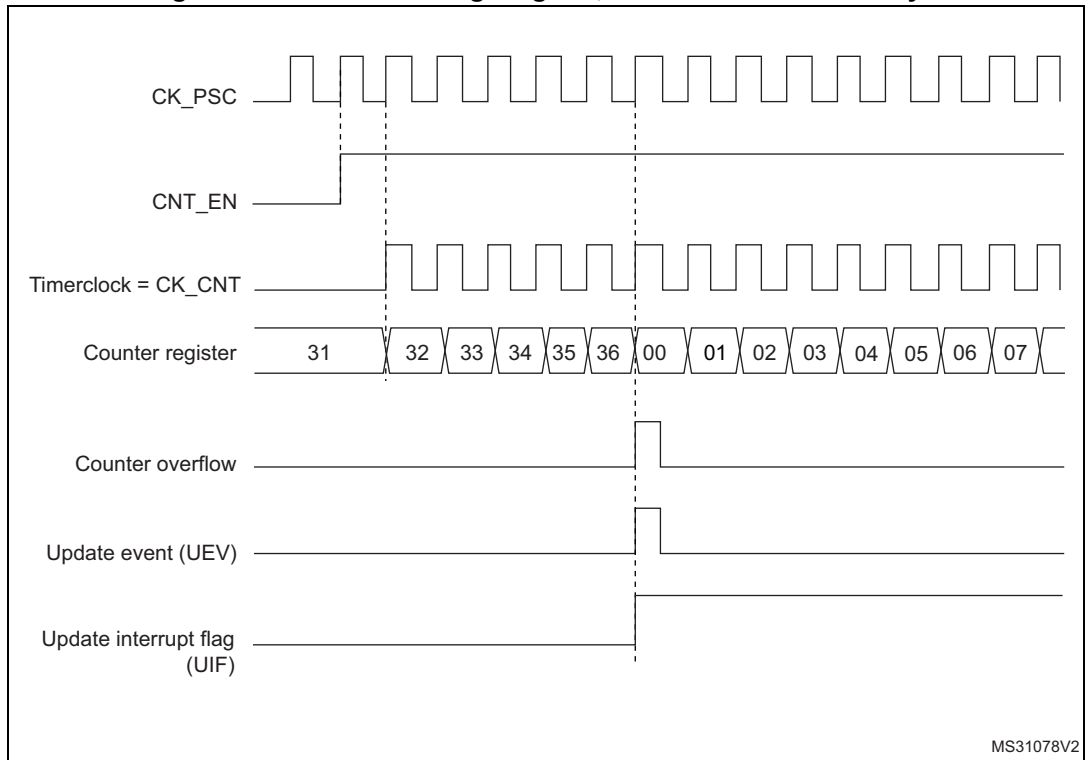
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as 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).

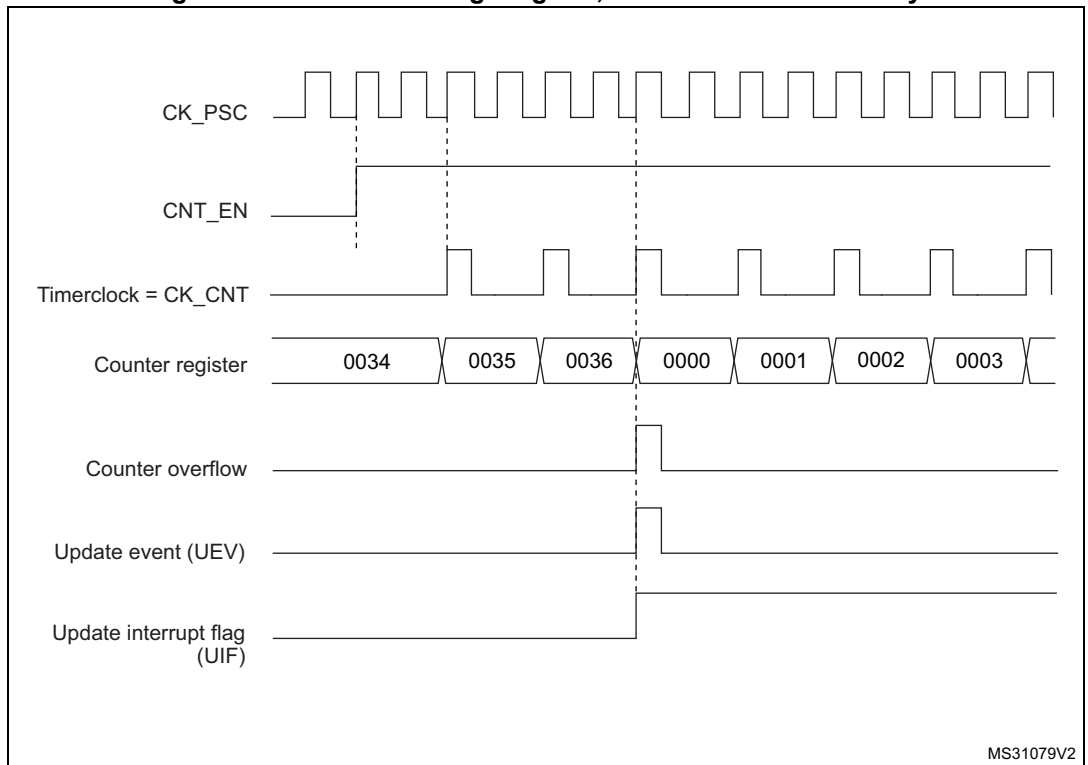
The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 104. Counter timing diagram, internal clock divided by 1**



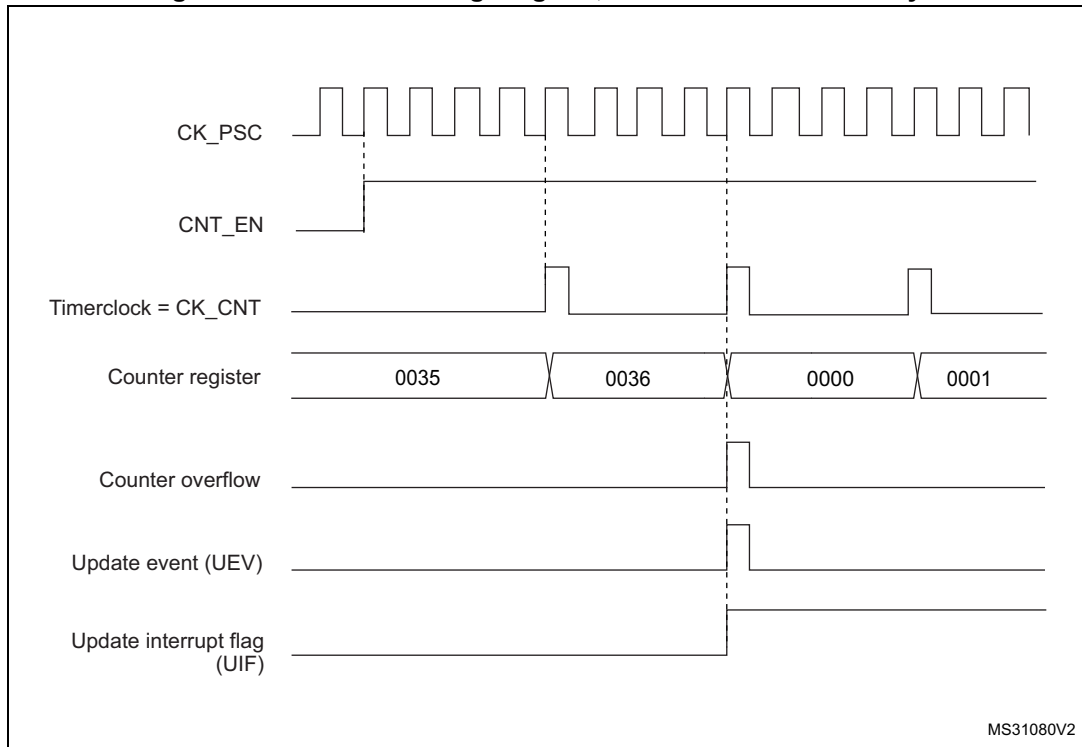
MS31078V2

**Figure 105. Counter timing diagram, internal clock divided by 2**



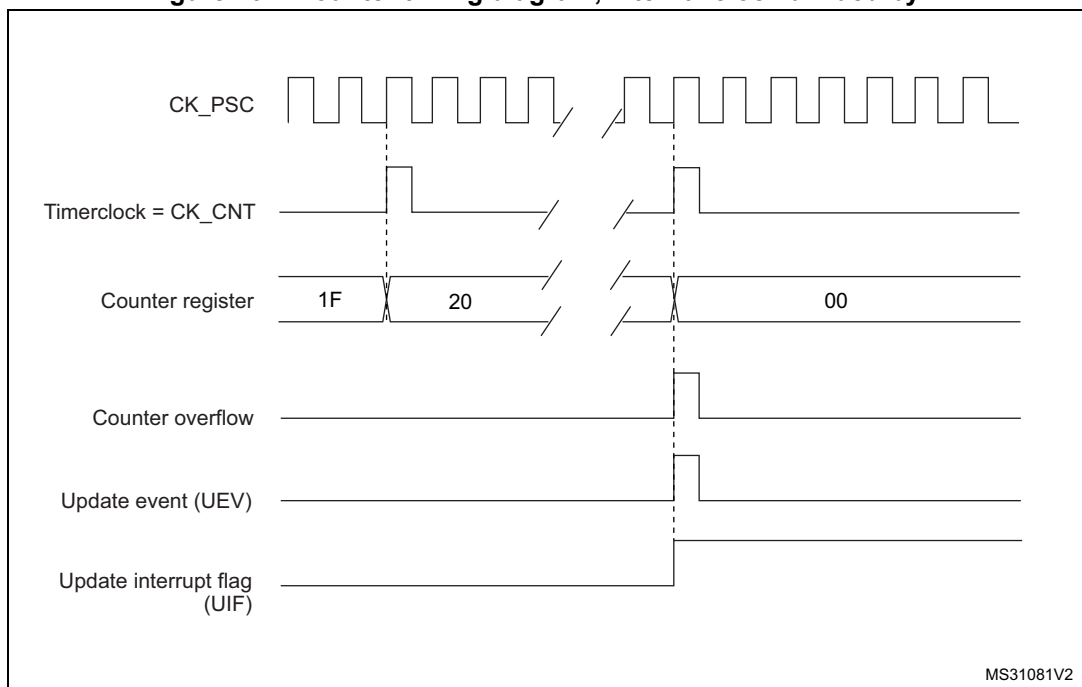
MS31079V2

Figure 106. Counter timing diagram, internal clock divided by 4



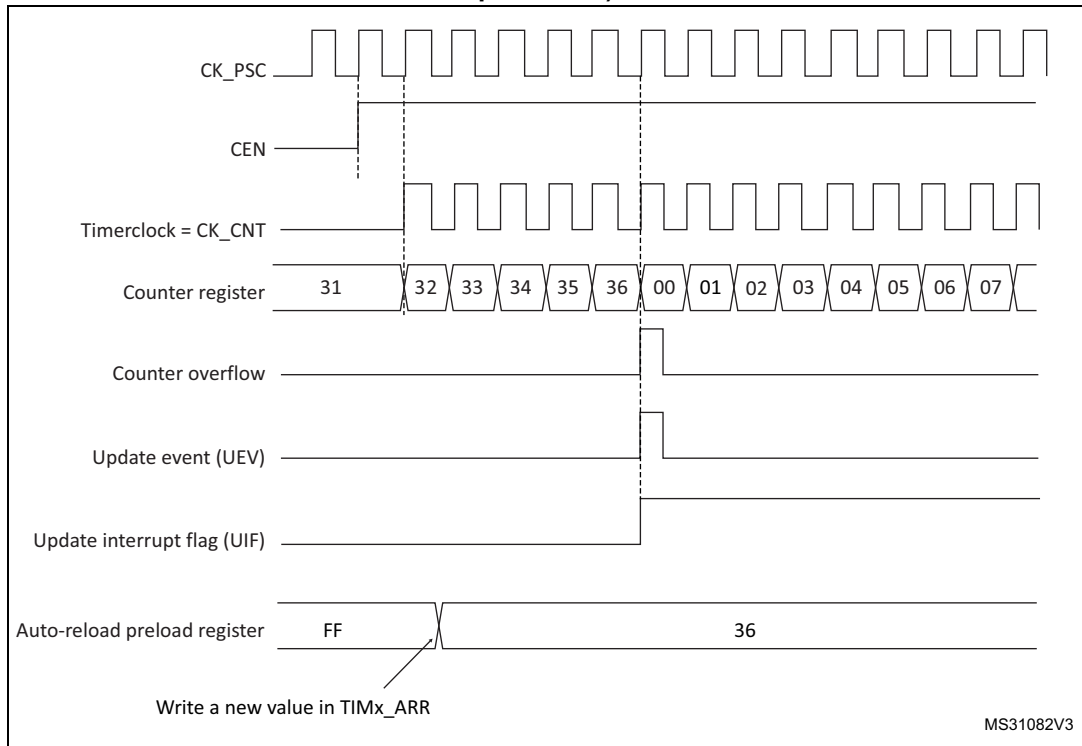
MS31080V2

Figure 107. Counter timing diagram, internal clock divided by N

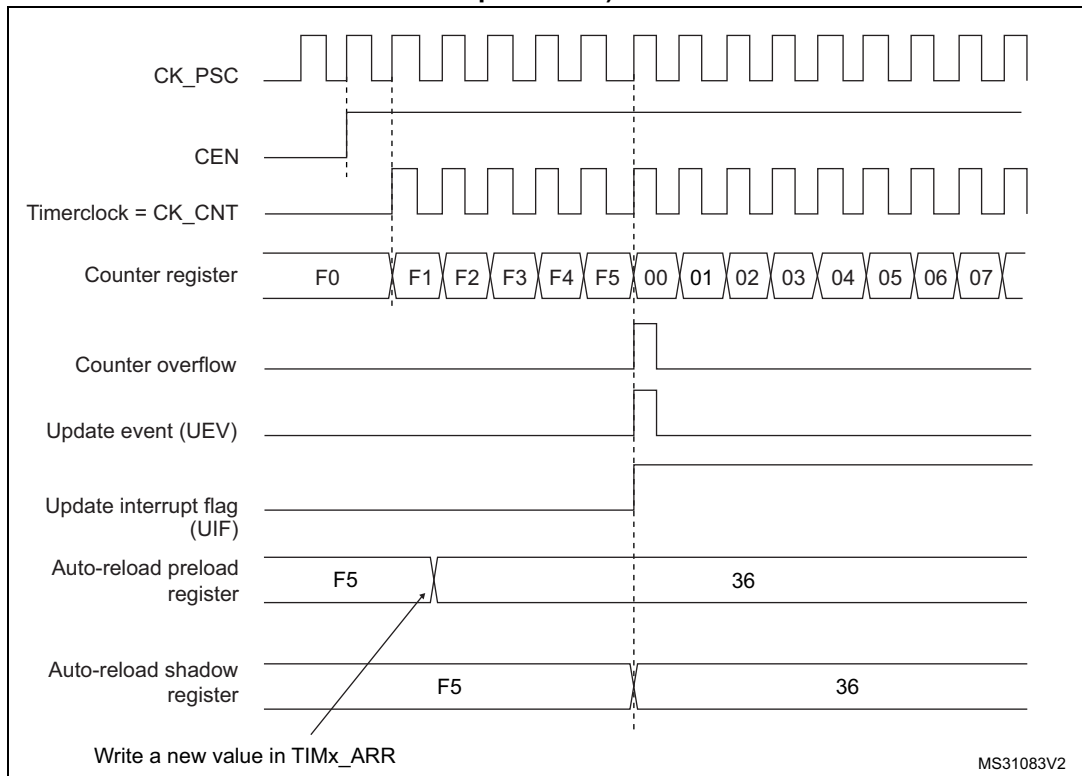


MS31081V2

**Figure 108. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 109. 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 110. Counter timing diagram, internal clock divided by 1

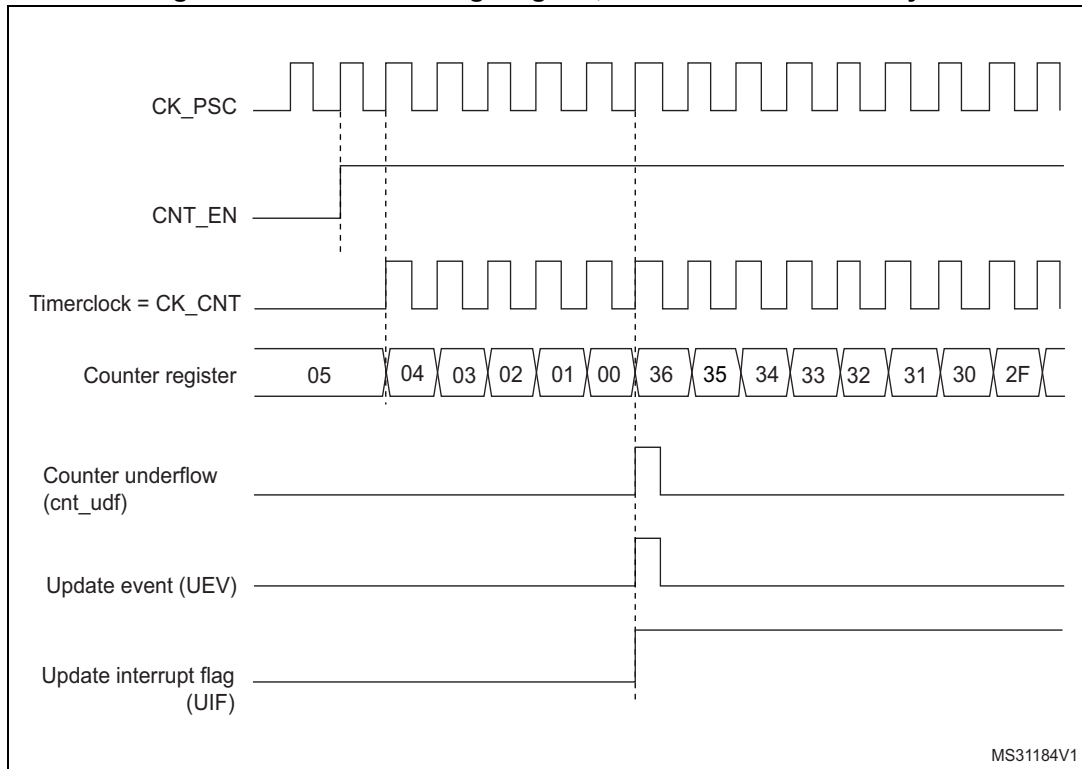


Figure 111. Counter timing diagram, internal clock divided by 2

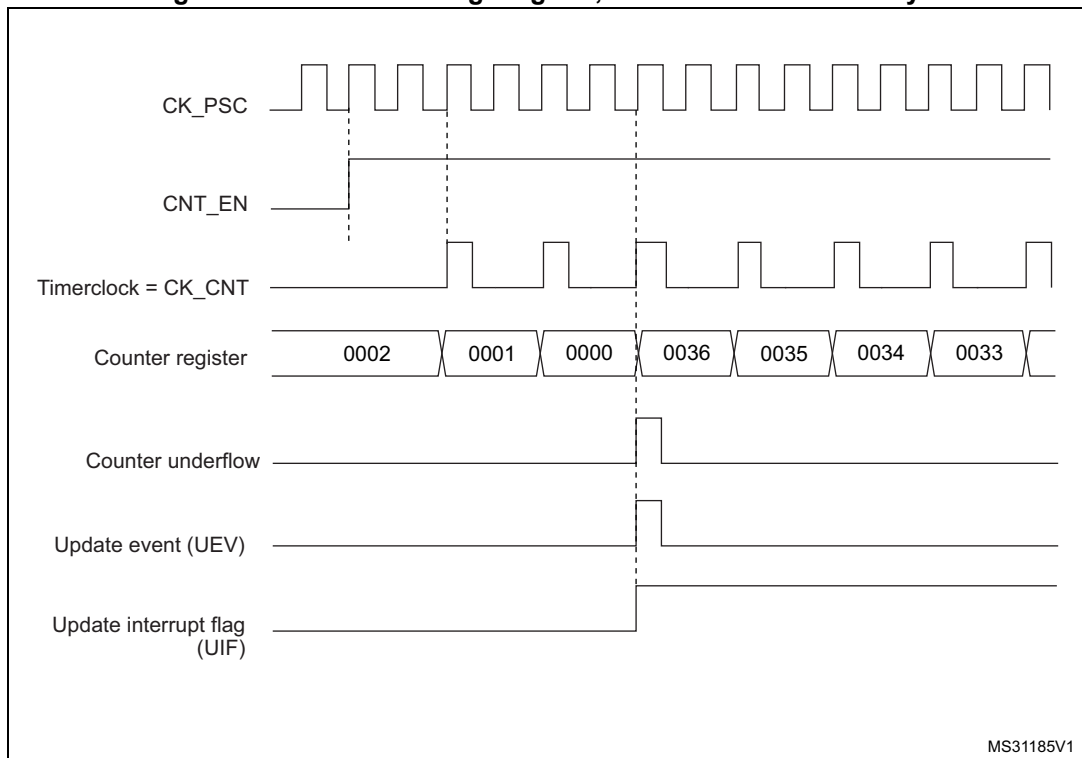


Figure 112. Counter timing diagram, internal clock divided by 4

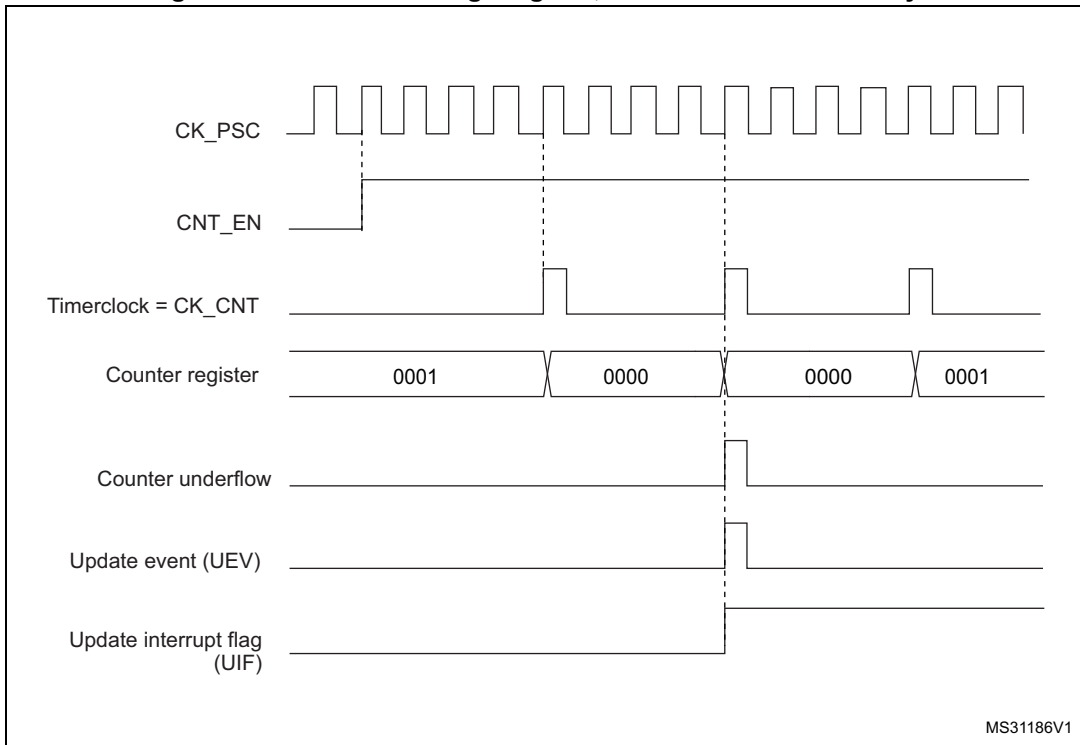
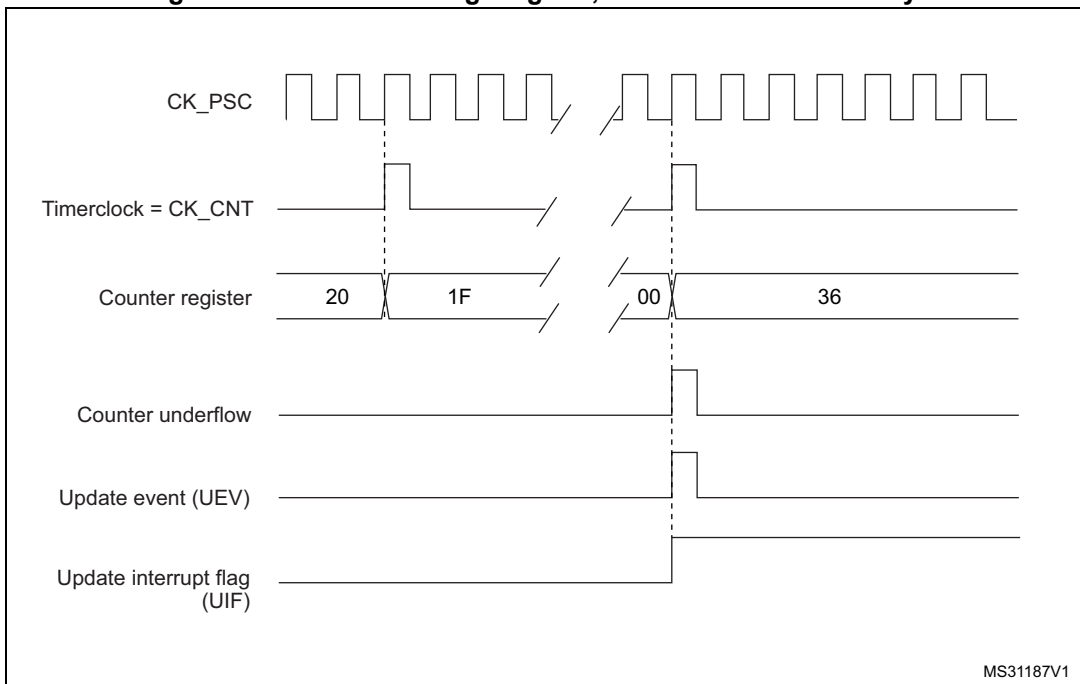
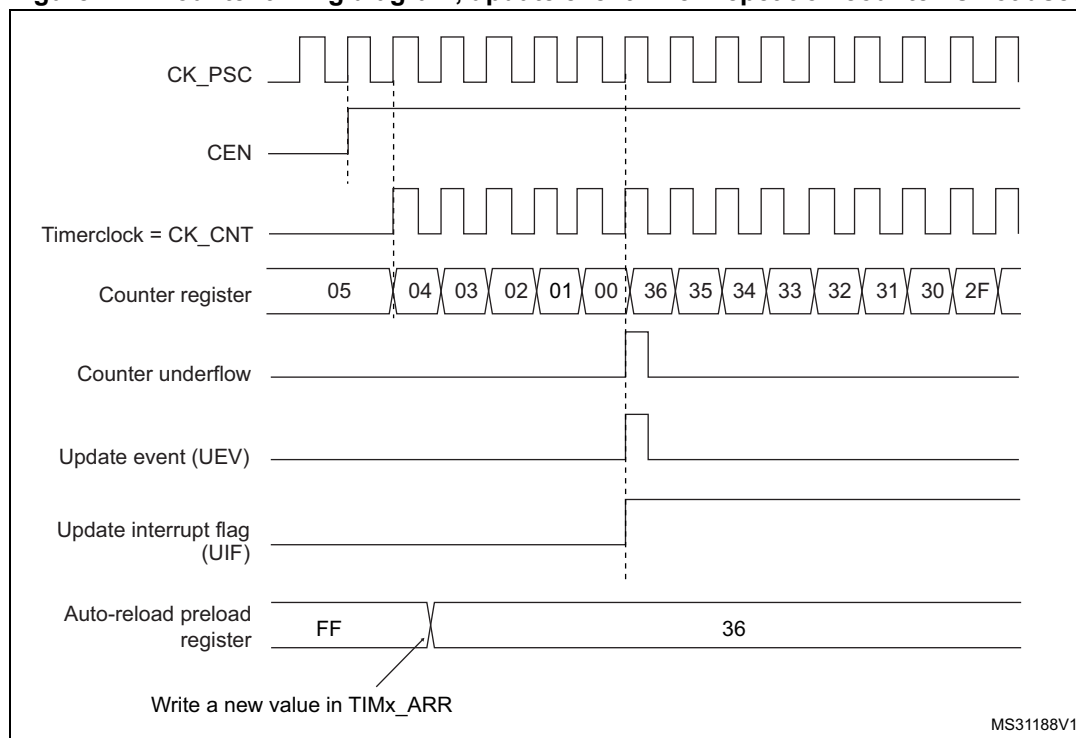


Figure 113. Counter timing diagram, internal clock divided by N



**Figure 114. Counter timing diagram, update event when repetition counter is not used**



**Center-aligned mode (up/down counting)**

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx\_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or

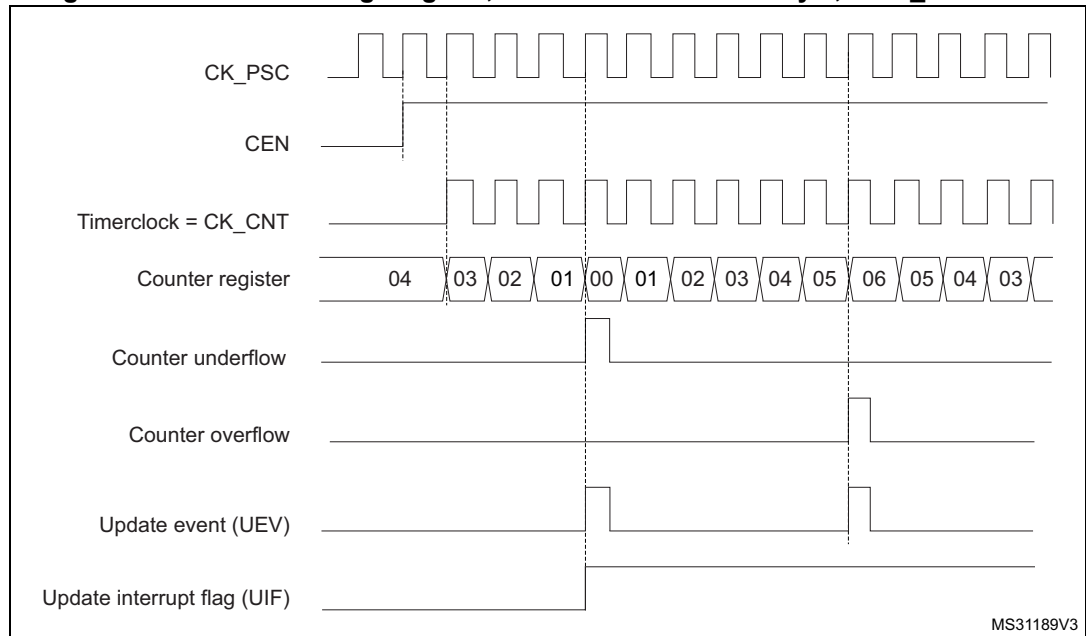
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx\_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

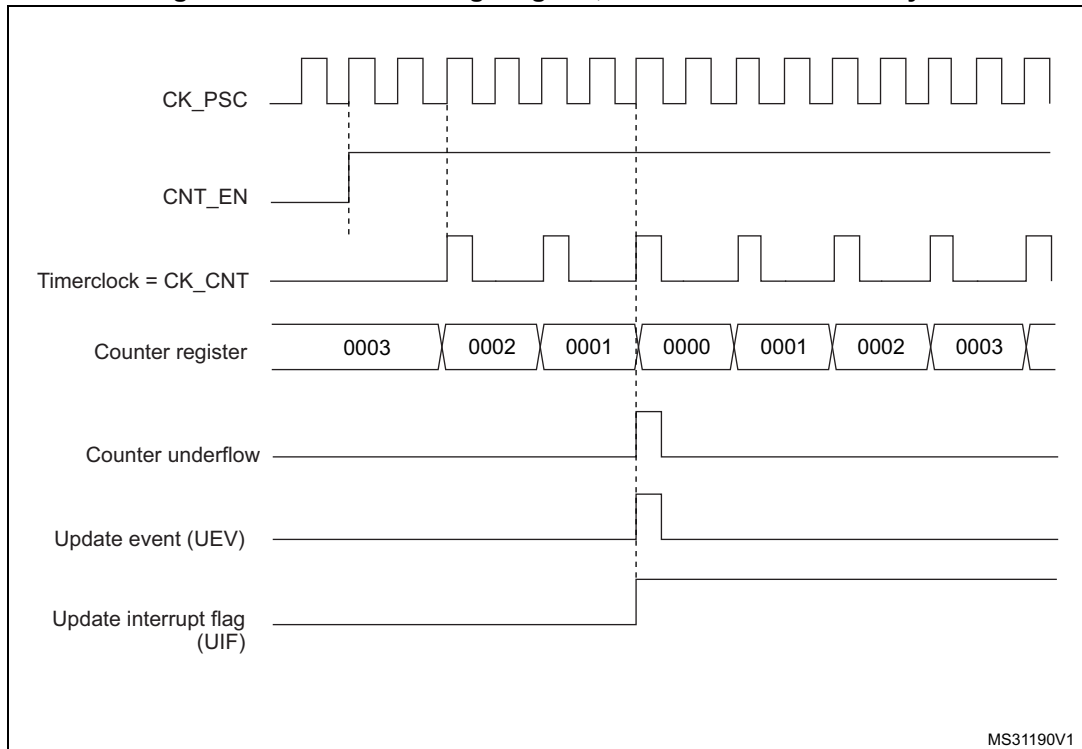
The following figures show some examples of the counter behavior for different clock frequencies.

**Figure 115. Counter timing diagram, internal clock divided by 1, TIMx\_ARR = 0x6**



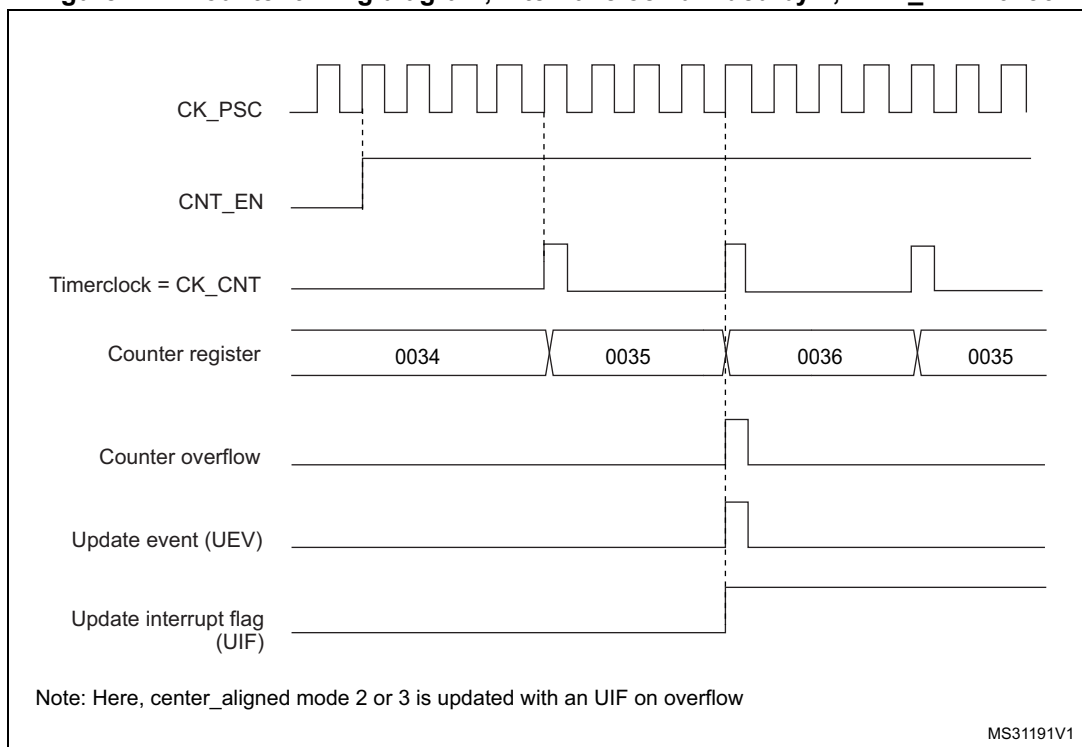
1. Here, center-aligned mode 1 is used (for more details refer to [Section 21.4: TIM1 registers](#)).

Figure 116. Counter timing diagram, internal clock divided by 2



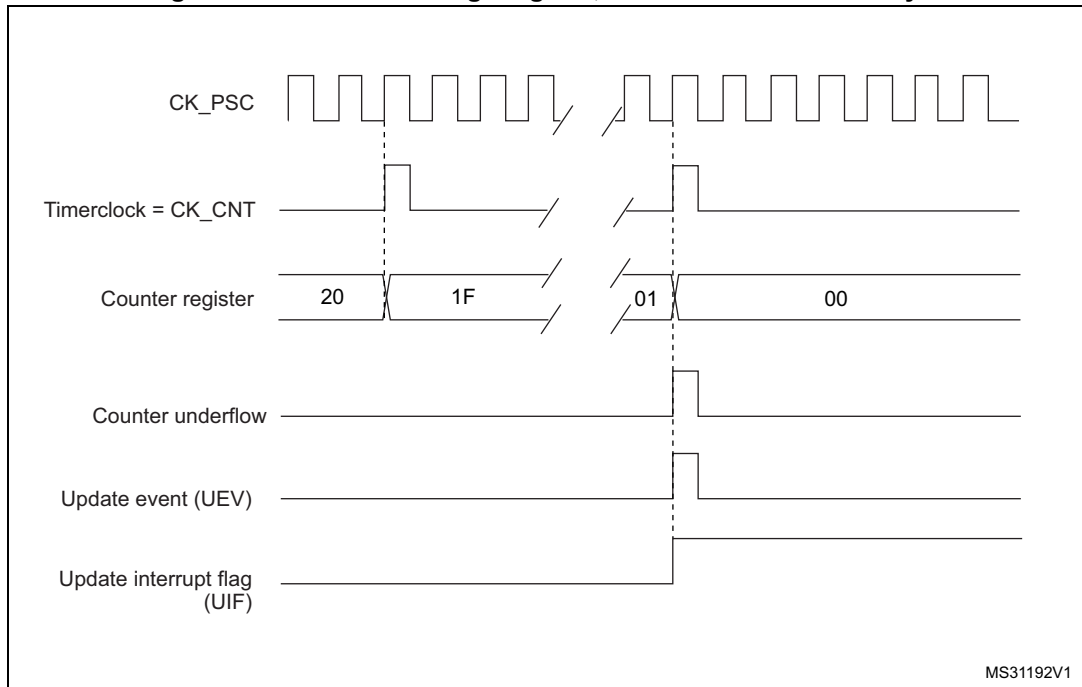
MS31190V1

Figure 117. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36



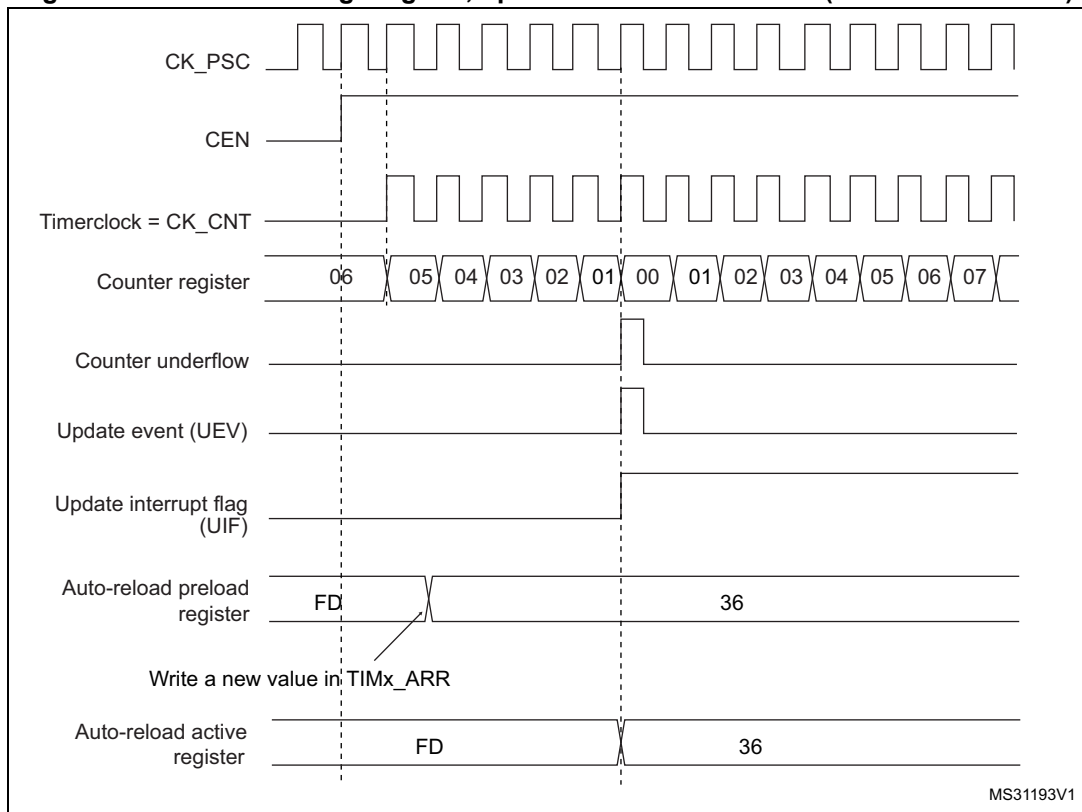
MS31191V1

Figure 118. Counter timing diagram, internal clock divided by N



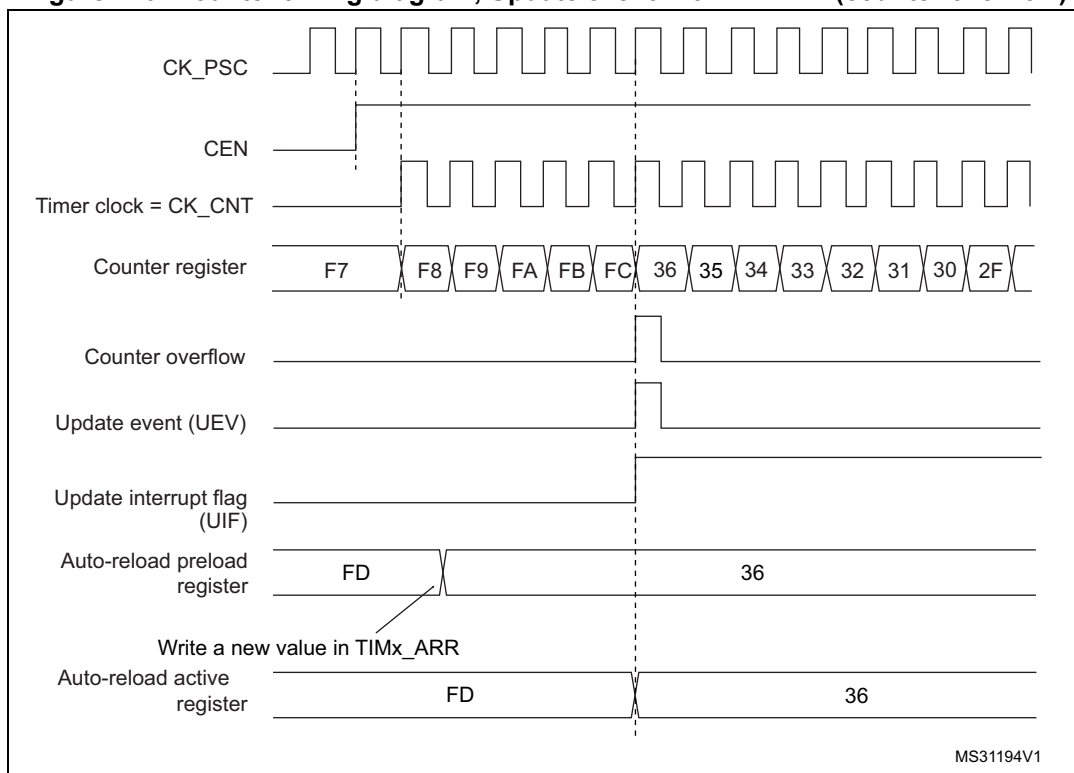
MS31192V1

Figure 119. Counter timing diagram, update event with ARPE=1 (counter underflow)



MS31193V1

Figure 120. Counter timing diagram, Update event with ARPE=1 (counter overflow)



### 21.3.3 Repetition counter

[Section 21.3.1: 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:

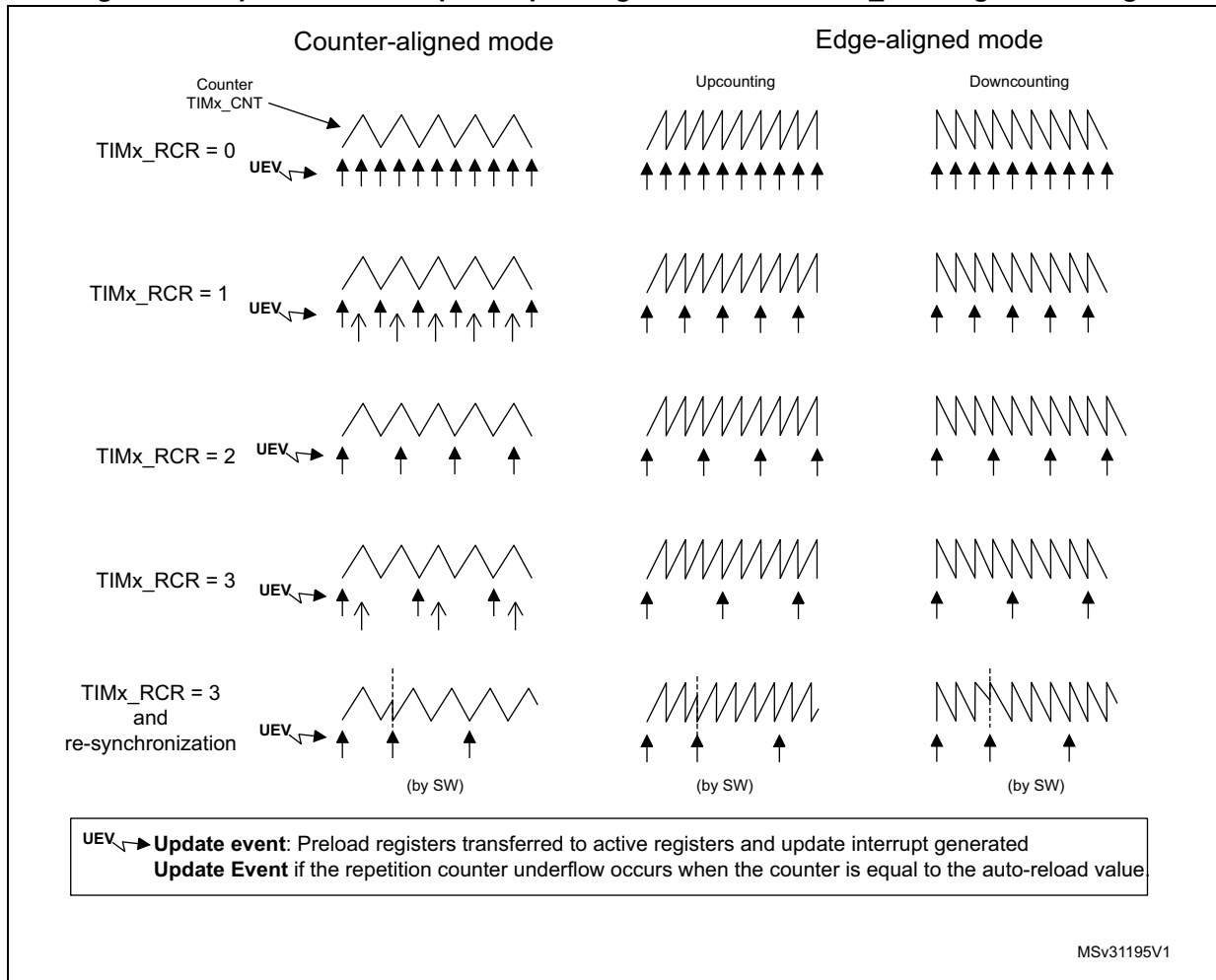
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is  $2xT_{ck}$ , due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to [Figure 121](#)). 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 121. Update rate examples depending on mode and TIMx\_RCR register settings**





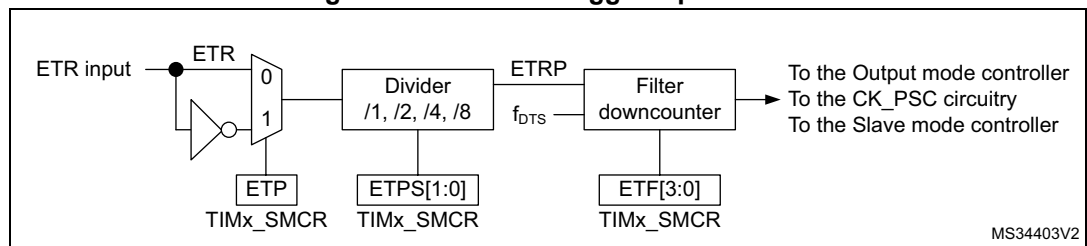
### 21.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see [Section 21.3.5](#))
- trigger for the slave mode (see [Section 21.3.26](#))
- PWM reset input for cycle-by-cycle current regulation (see [Section 21.3.7](#))

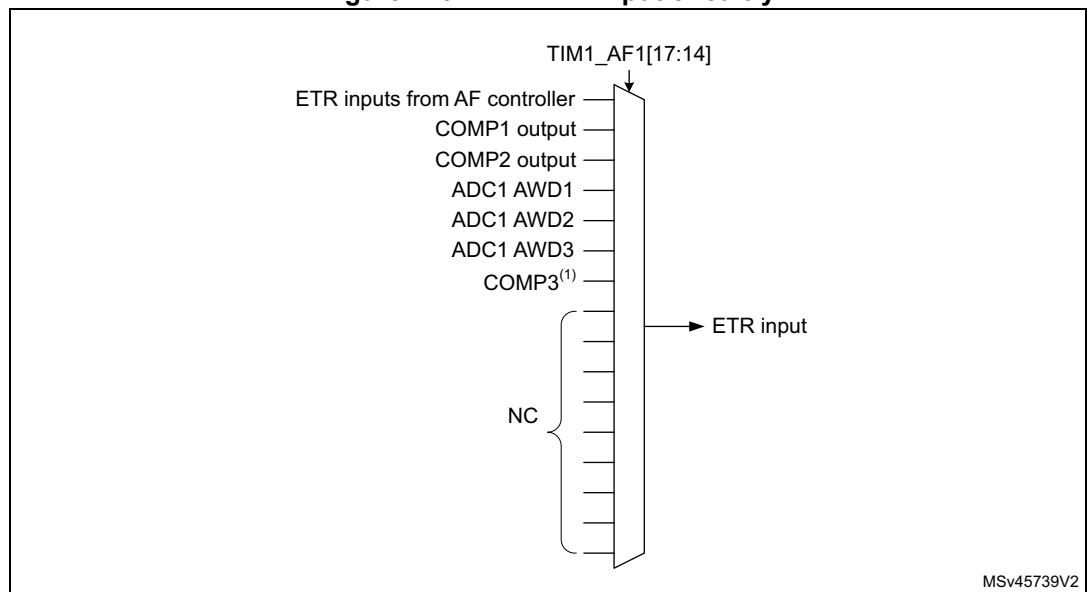
[Figure 122](#) below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

**Figure 122. External trigger input block**



The ETR input comes from multiple sources: input pins (default configuration), comparator outputs and analog watchdogs. The selection is done with the ETRSEL[3:0] bitfield.

**Figure 123. TIM1 ETR input circuitry**



1. Available on STM32G0B1xx and STM32G0C1xx saletypes only.

### 21.3.5 Clock selection

The counter clock can be provided by the following clock sources:

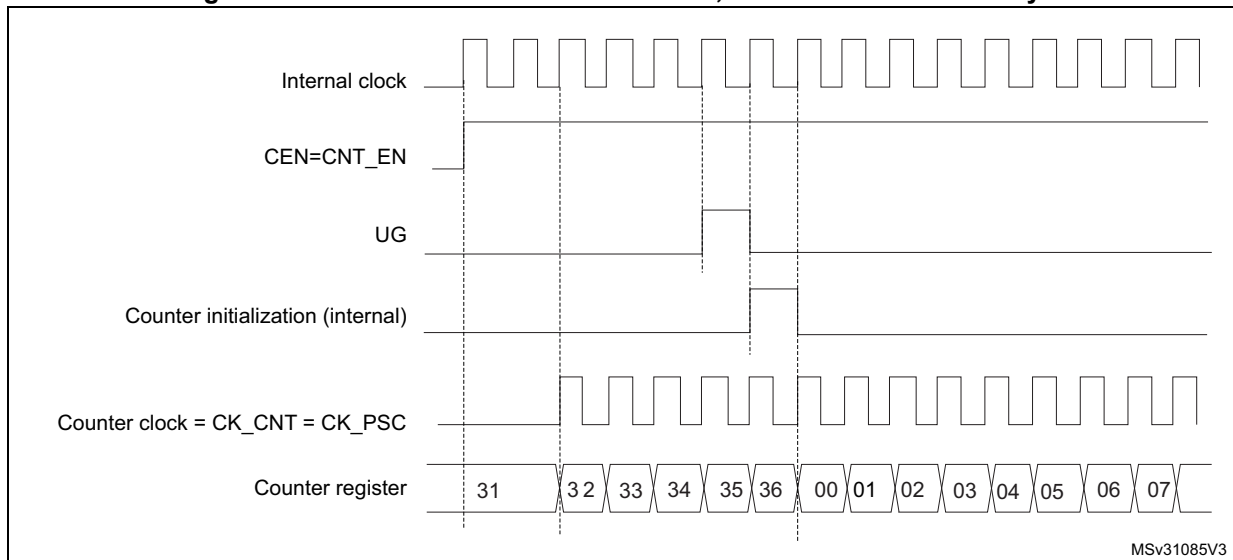
- Internal clock (CK\_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

#### 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 124 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

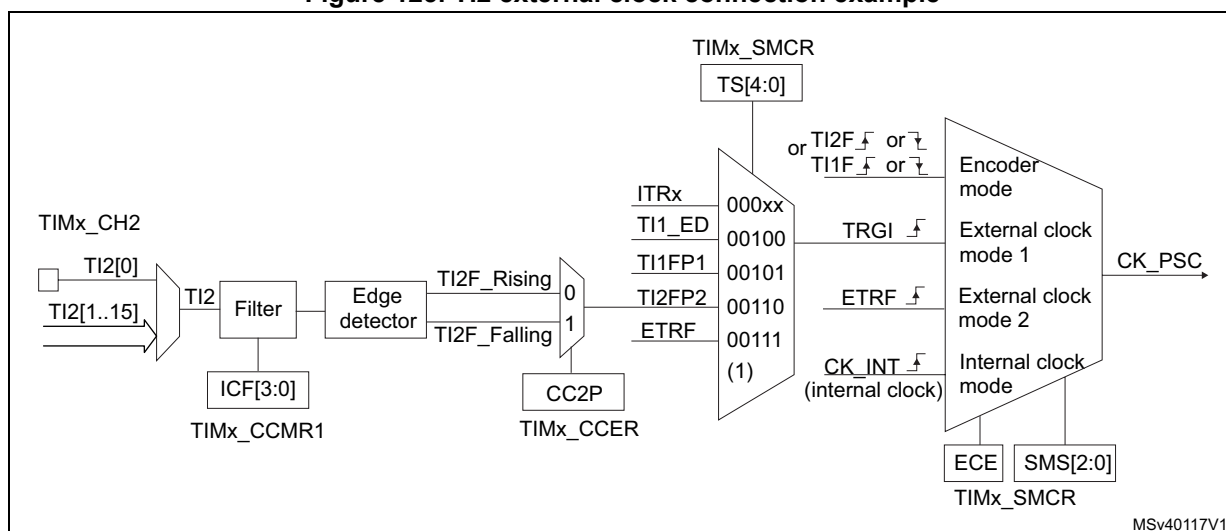
Figure 124. Control circuit in normal mode, internal clock divided by 1



#### 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 125. T12 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 T12 input, use the following procedure:

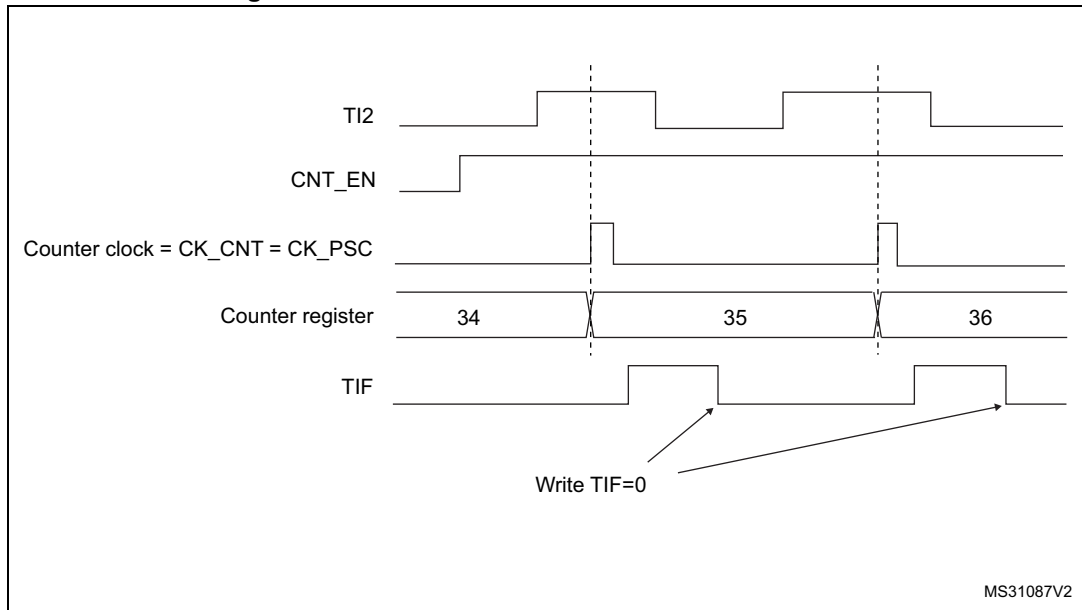
1. Select the proper T12x source (internal or external) with the T12SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the T12 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the 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 T12 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 T12, the counter counts once and the TIF flag is set.

The delay between the rising edge on T12 and the actual clock of the counter is due to the resynchronization circuit on T12 input.

Figure 126. Control circuit in external clock mode 1



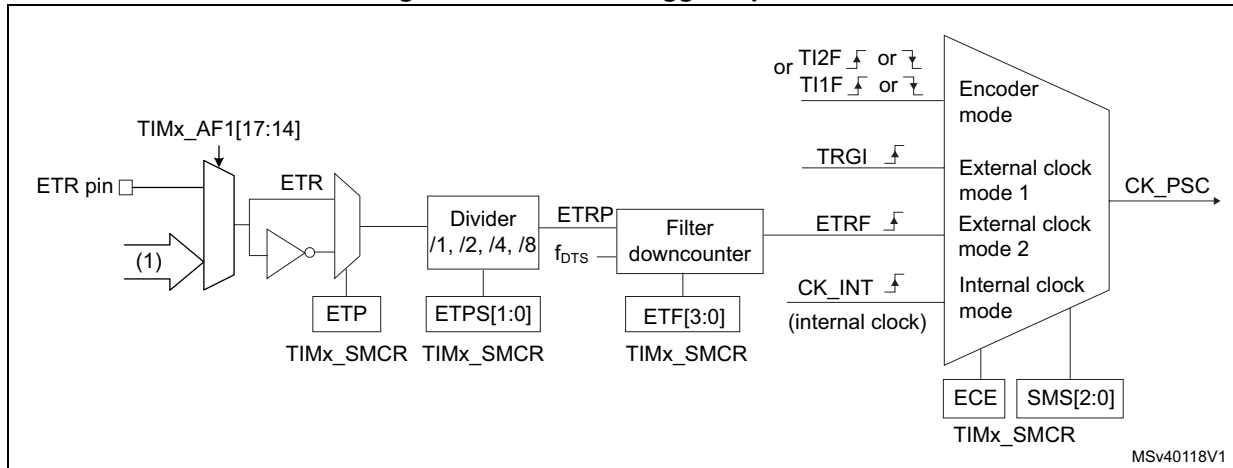
**External clock source mode 2**

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 127](#) gives an overview of the external trigger input block.

Figure 127. External trigger input block



1. Refer to [Figure 123: TIM1 ETR input circuitry](#).

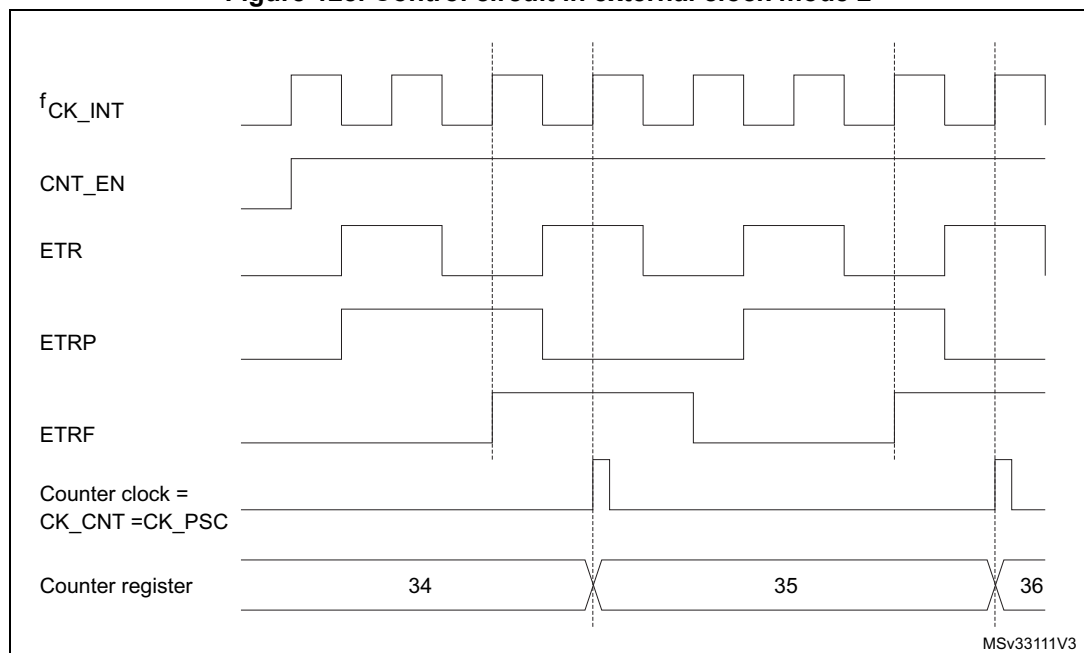
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETRF[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 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 128. Control circuit in external clock mode 2**



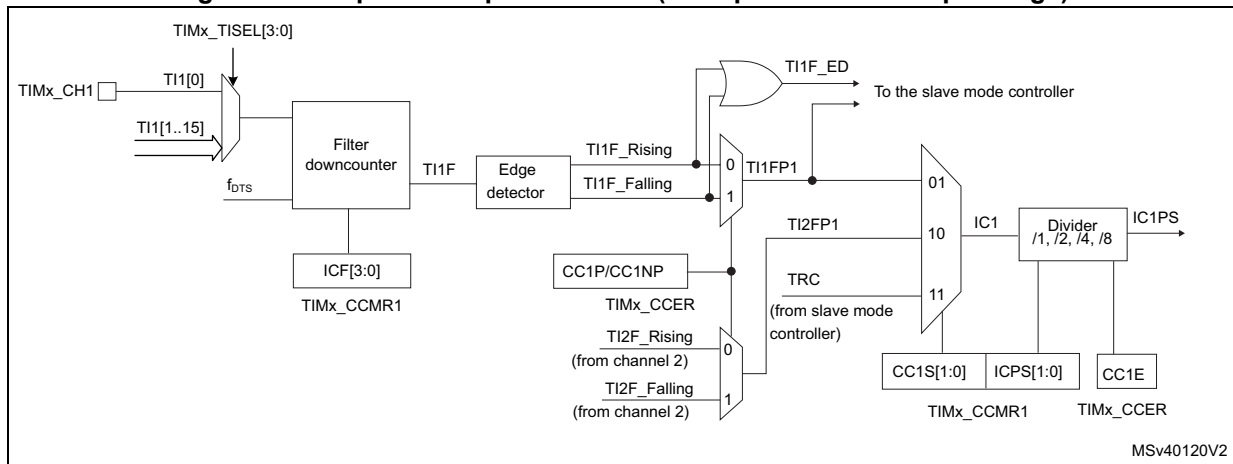
### 21.3.6 Capture/compare channels

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 129 to Figure 132 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 129. 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 130. Capture/compare channel 1 main circuit

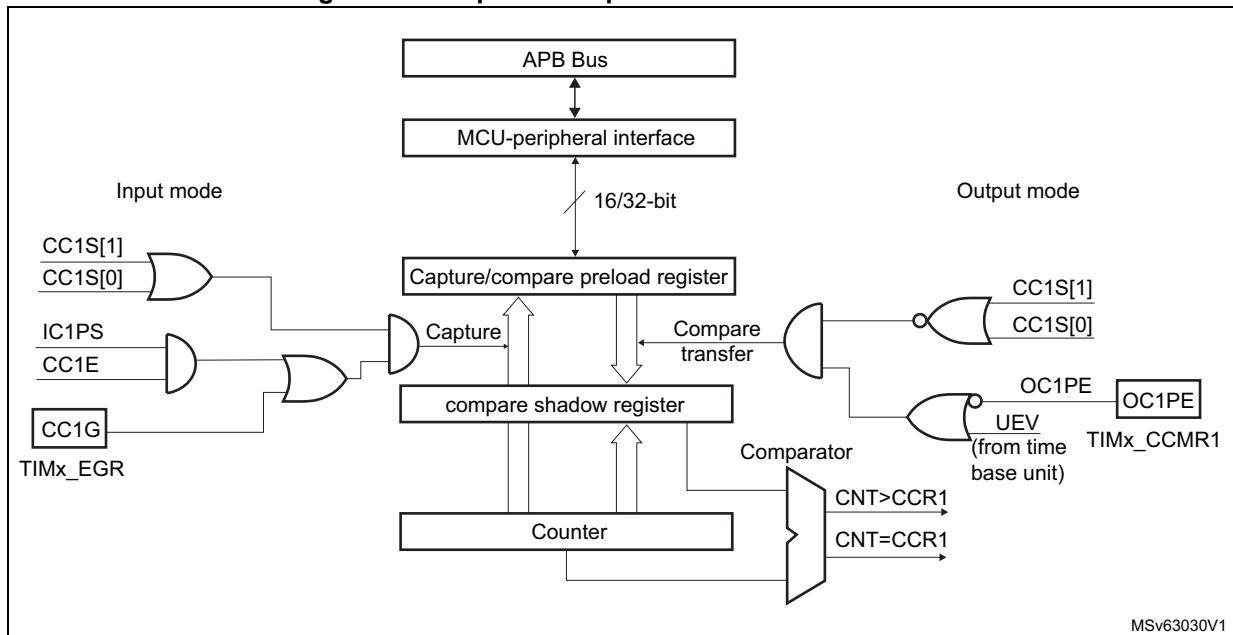
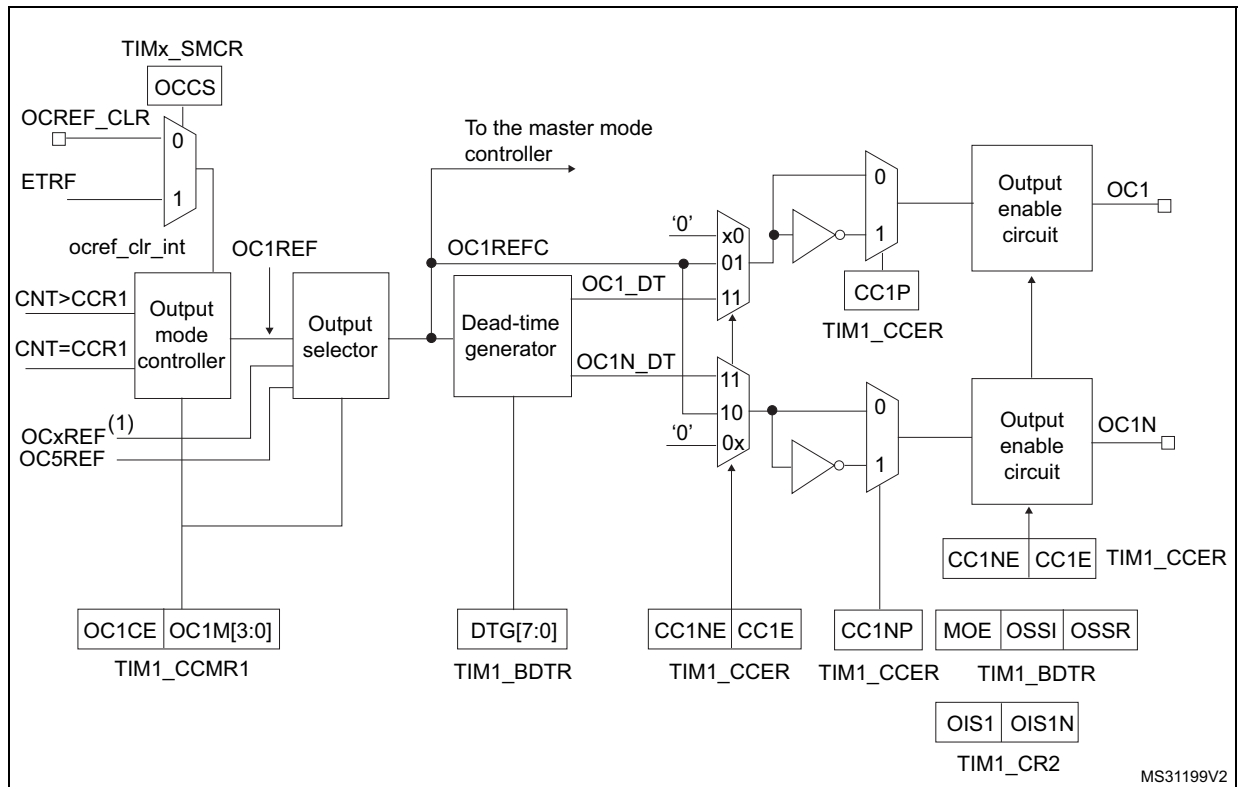


Figure 131. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 132. Output stage of capture/compare channel (channel 4)

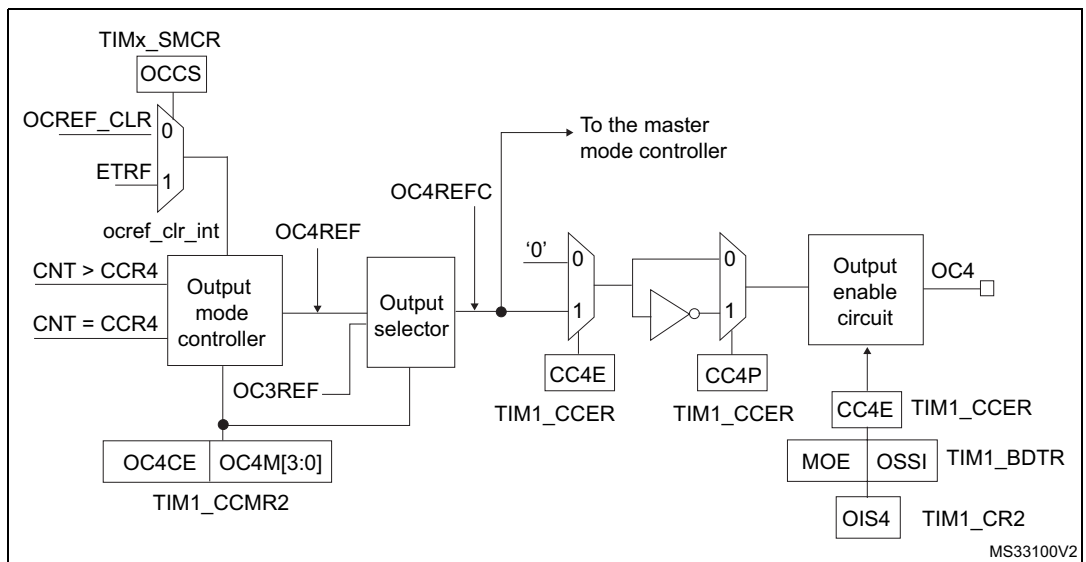
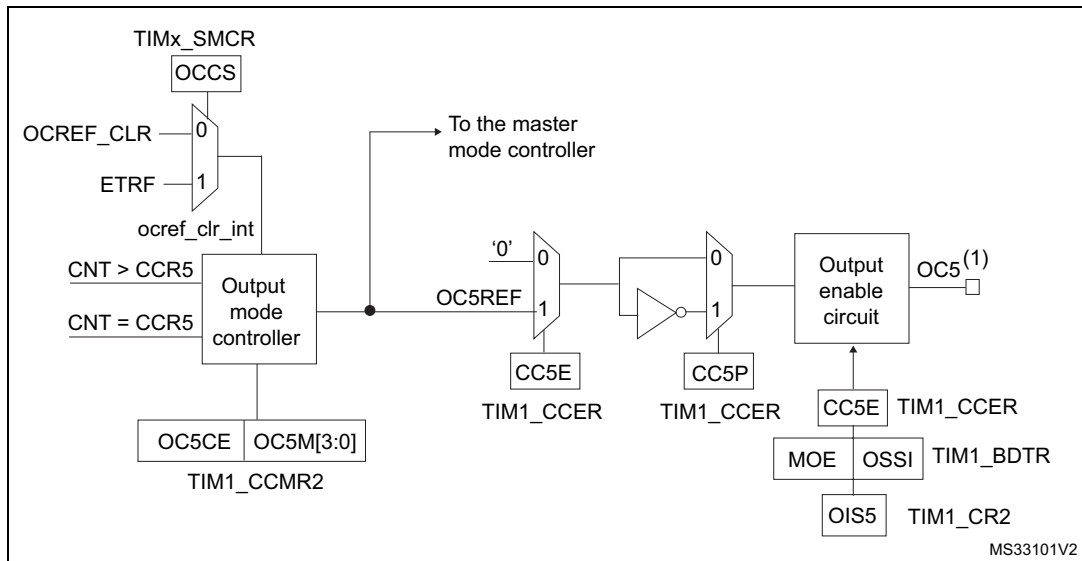


Figure 133. Output stage of capture/compare channel (channel 5, idem ch. 6)



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.

### 21.3.7 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.

### 21.3.8 PWM input mode

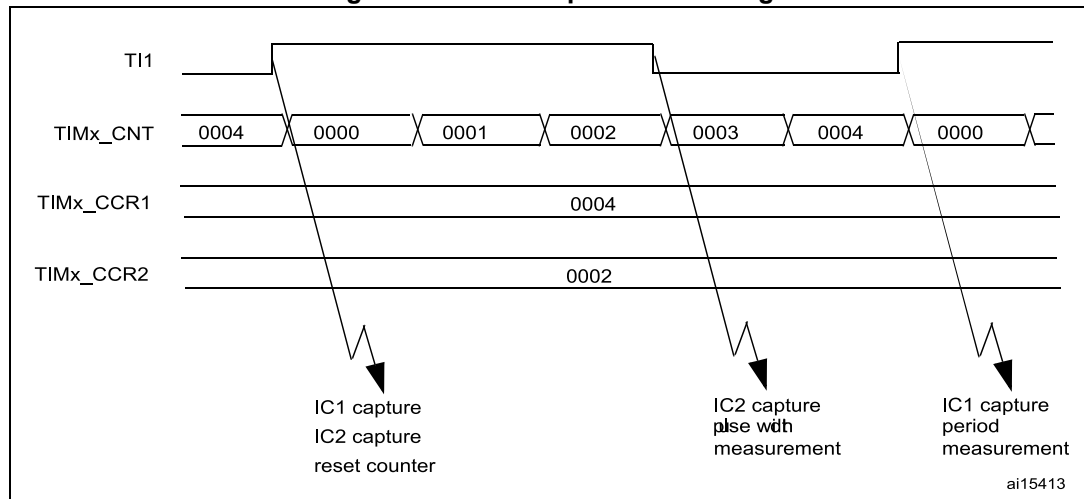
This mode is a particular case of input capture mode. The procedure is the same except:

- 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 134. PWM input mode timing



### 21.3.9 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.

### 21.3.10 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.

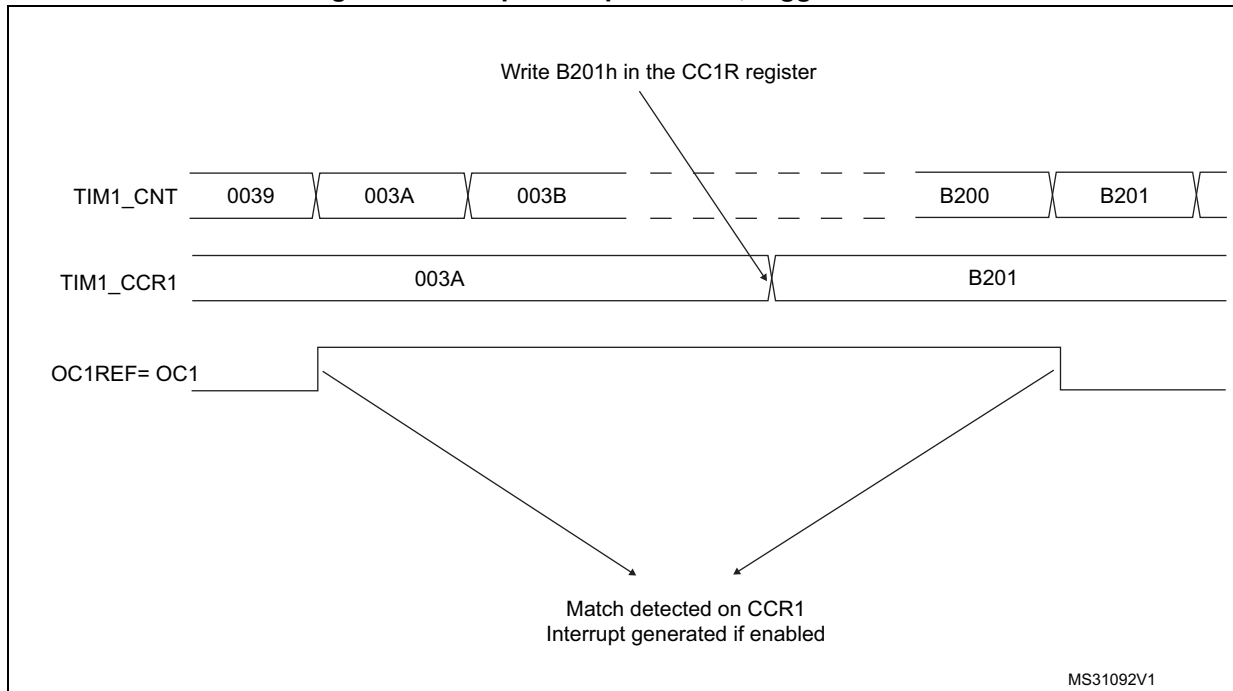
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

#### 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 135](#).

Figure 135. Output compare mode, toggle on OC1



### 21.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $TIMx\_CCRx \leq TIMx\_CNT$  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**

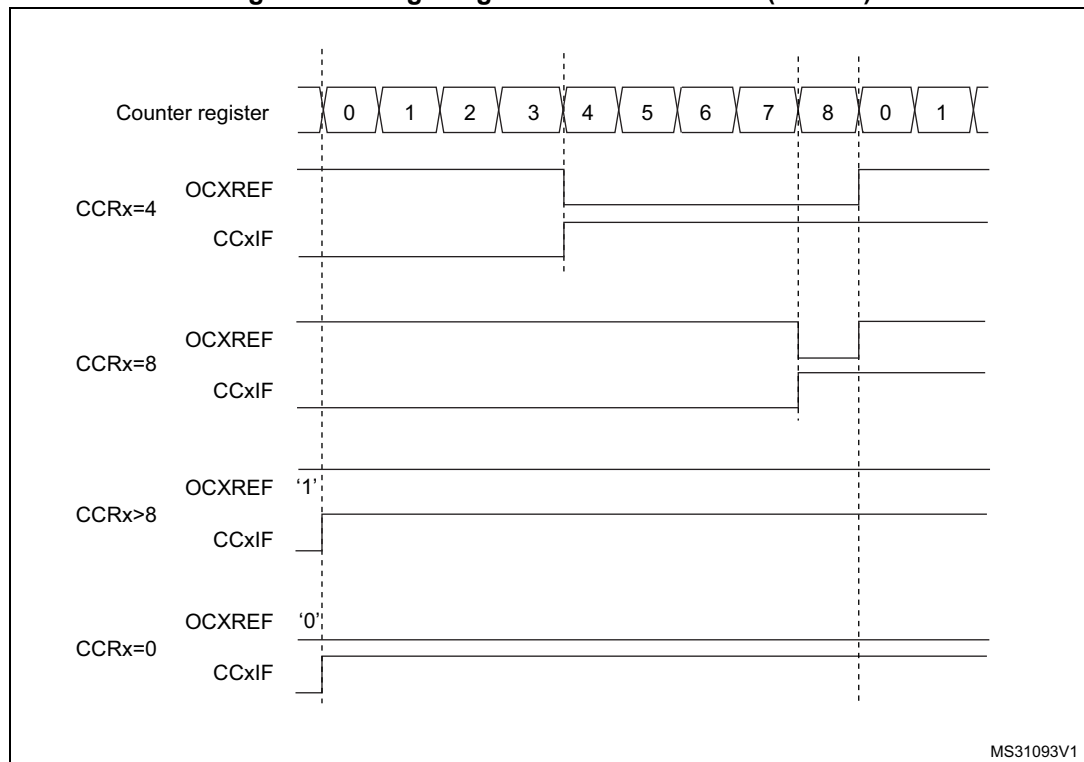
- Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to the [Upcounting mode on page 520](#).

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 136](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 136. 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 524](#)

In PWM mode 1, the reference signal OCxRef is low as long as TIMx\_CNT > TIMx\_CCRx else it becomes high. If the compare value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

**PWM center-aligned mode**

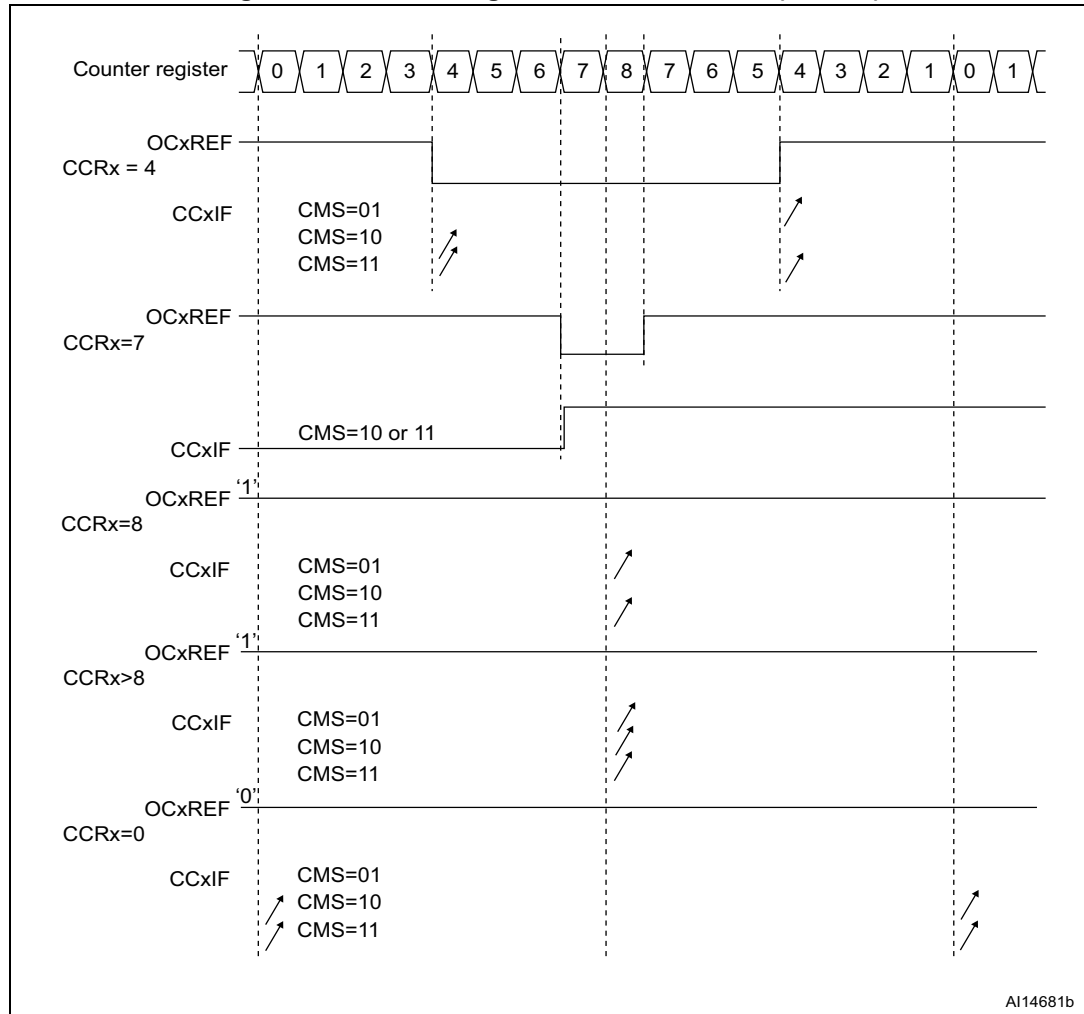
Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are different from '00' (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx\_CR1 register is updated by hardware and must not be changed by software. Refer to the [Center-aligned mode \(up/down counting\) on page 527](#).

Figure 137 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 137. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx\_CNT > TIMx\_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx\_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx\_EGR register) just before starting the counter and not to write the counter while it is running.

### 21.3.12 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

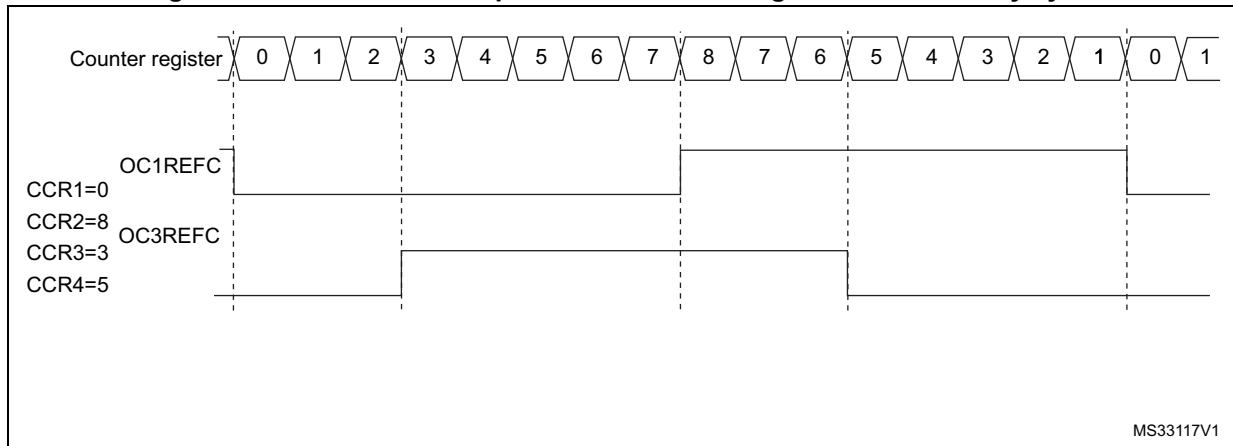
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

*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 138](#) 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 138. Generation of 2 phase-shifted PWM signals with 50% duty cycle



### 21.3.13 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).

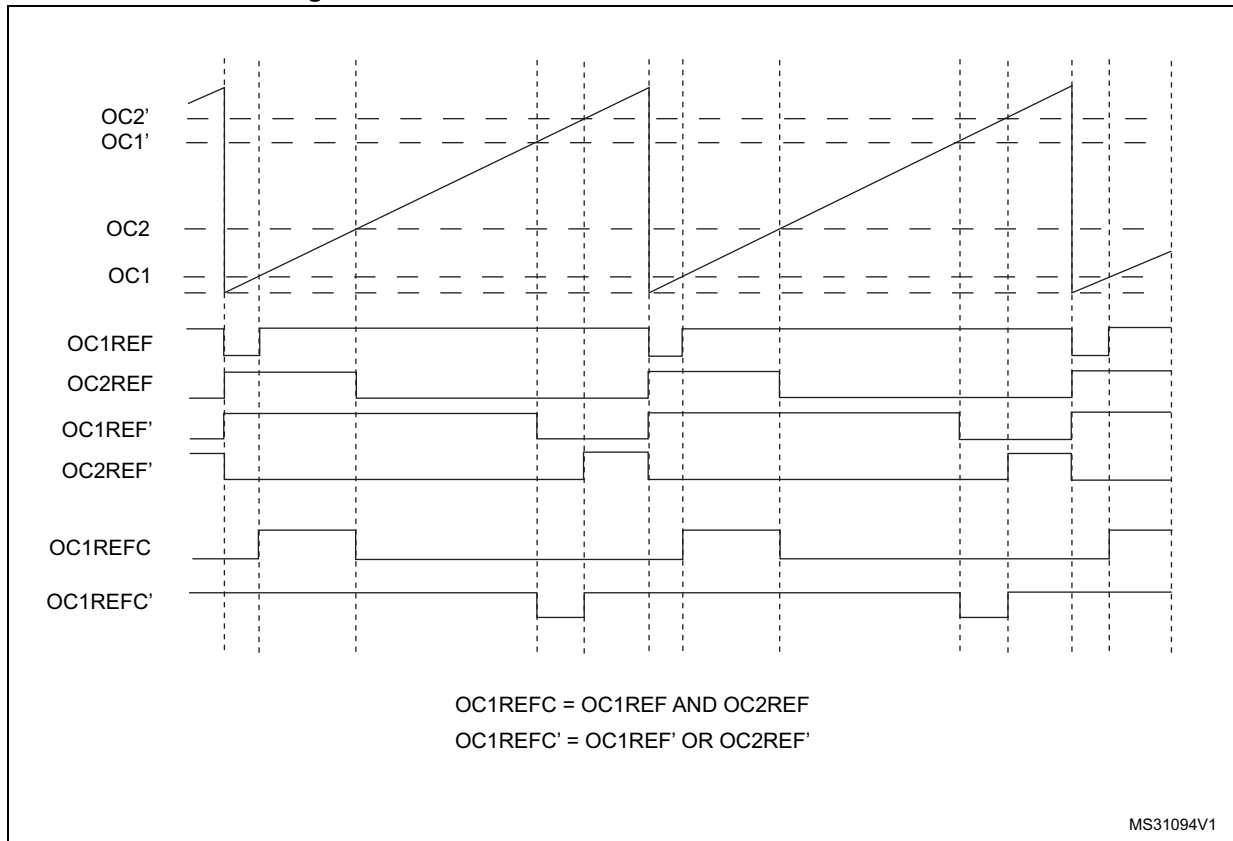
*Note:* The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 139 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 139. Combined PWM mode on channel 1 and 3



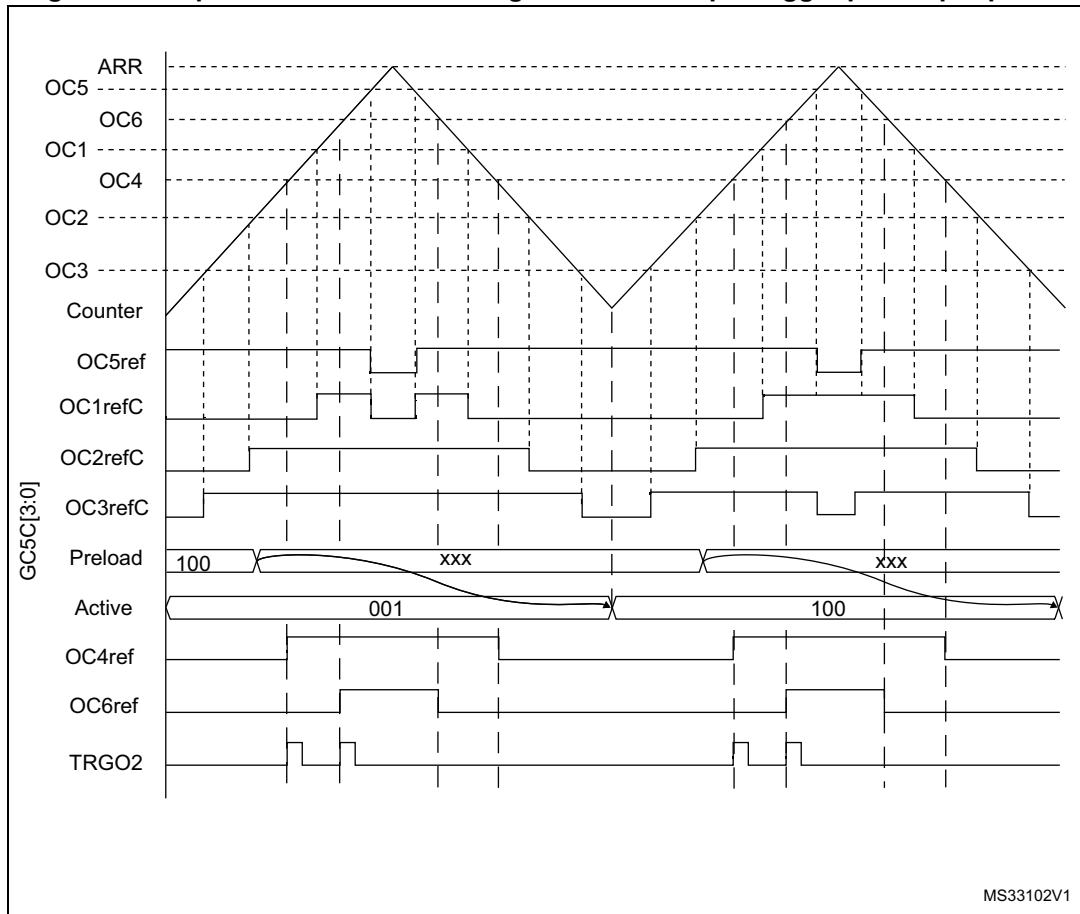
### 21.3.14 Combined 3-phase PWM mode

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 140. 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 21.3.27: ADC synchronization](#) for more details.

### 21.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx\_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx\_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx\_BDTR and TIMx\_CR2 registers. Refer to [Table 120: Output control bits for complementary OCx and OCxN channels with break feature on page 596](#) for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

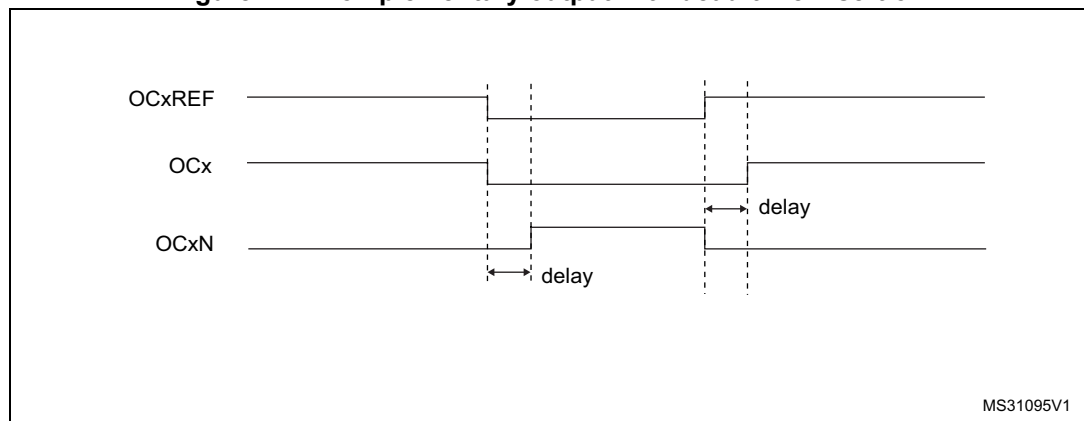
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

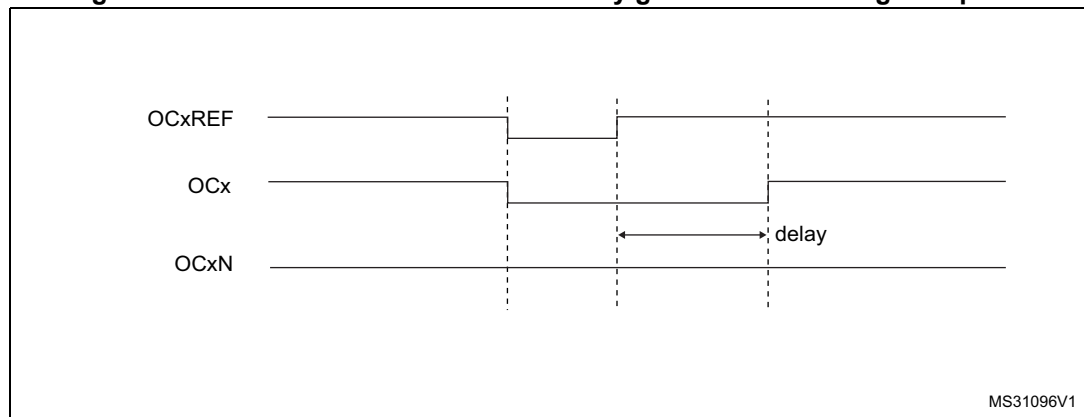
If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

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)

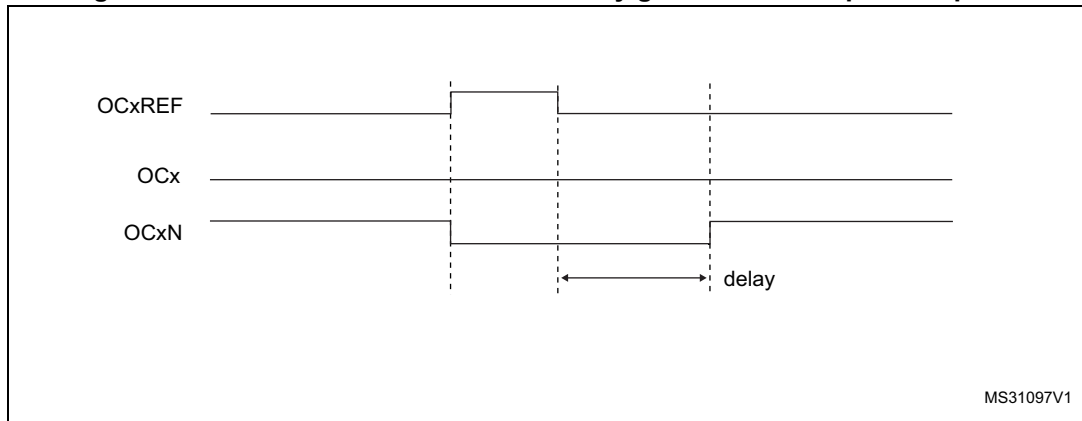
**Figure 141. Complementary output with dead-time insertion**



**Figure 142. Dead-time waveforms with delay greater than the negative pulse**



**Figure 143. 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 21.4.20: 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.*

**21.3.16 Using the break function**

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 timer. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx\_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx\_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx\_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to [Table 120: Output control bits for complementary OCx and OCxN channels with break feature on page 596](#) 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<sup>®</sup>-M0+ LOCKUP output
  - the PVD output
  - the SRAM parity error signal
  - a flash memory ECC dual error detection
  - a clock failure event generated by the CSS detector
  - the output from a comparator, with polarity selection and optional digital filtering

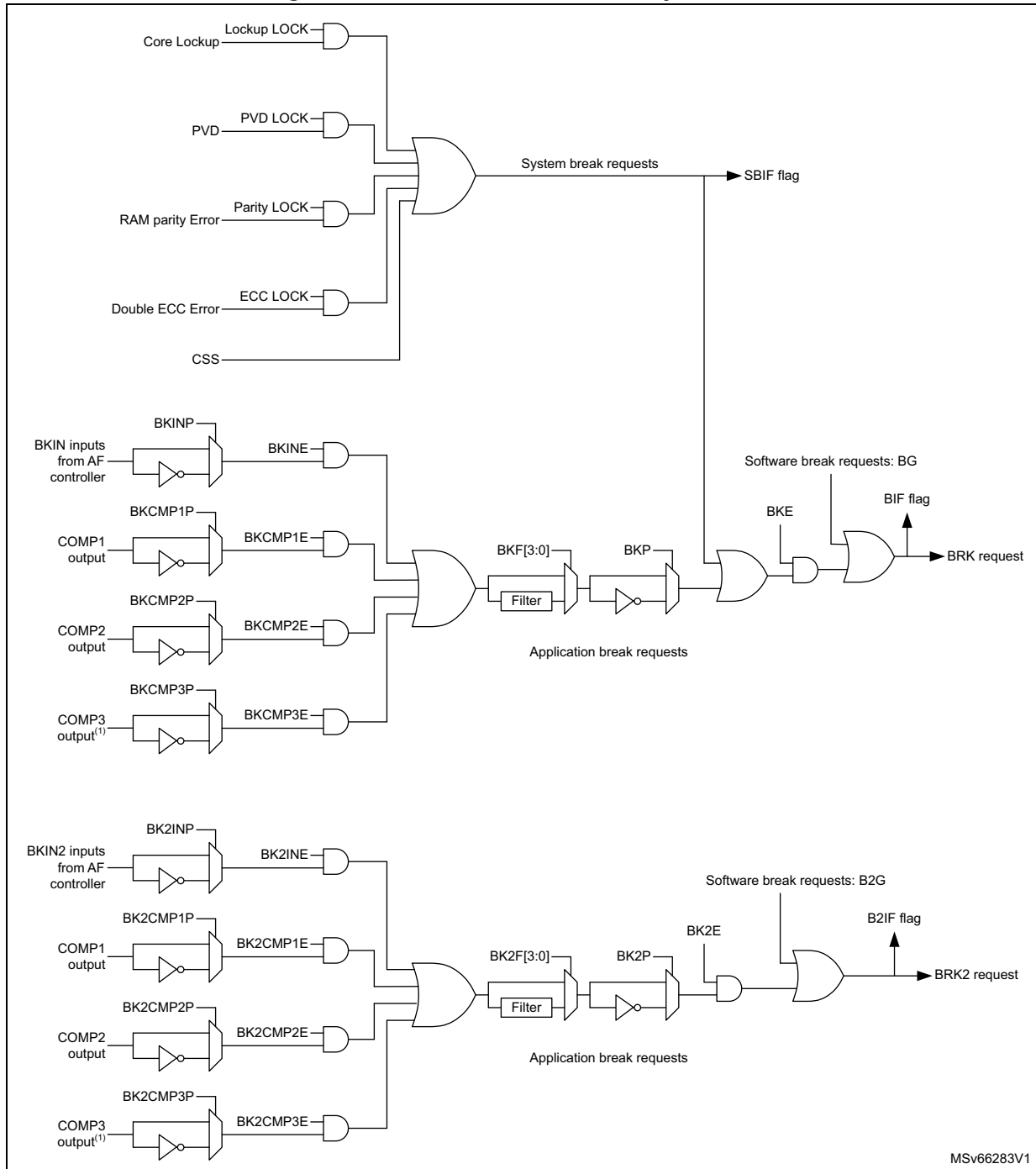
The sources for break2 (BRK2) are:

- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source coming from a comparator output.

Break events can also be generated by software using BG and B2G bits in the TIMx\_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per [Figure 144](#) below.

**Figure 144. Break and Break2 circuitry overview**



1. Available on STM32G0B1xx and STM32G0C1xx salestypes only.

**Note:** An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the 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.

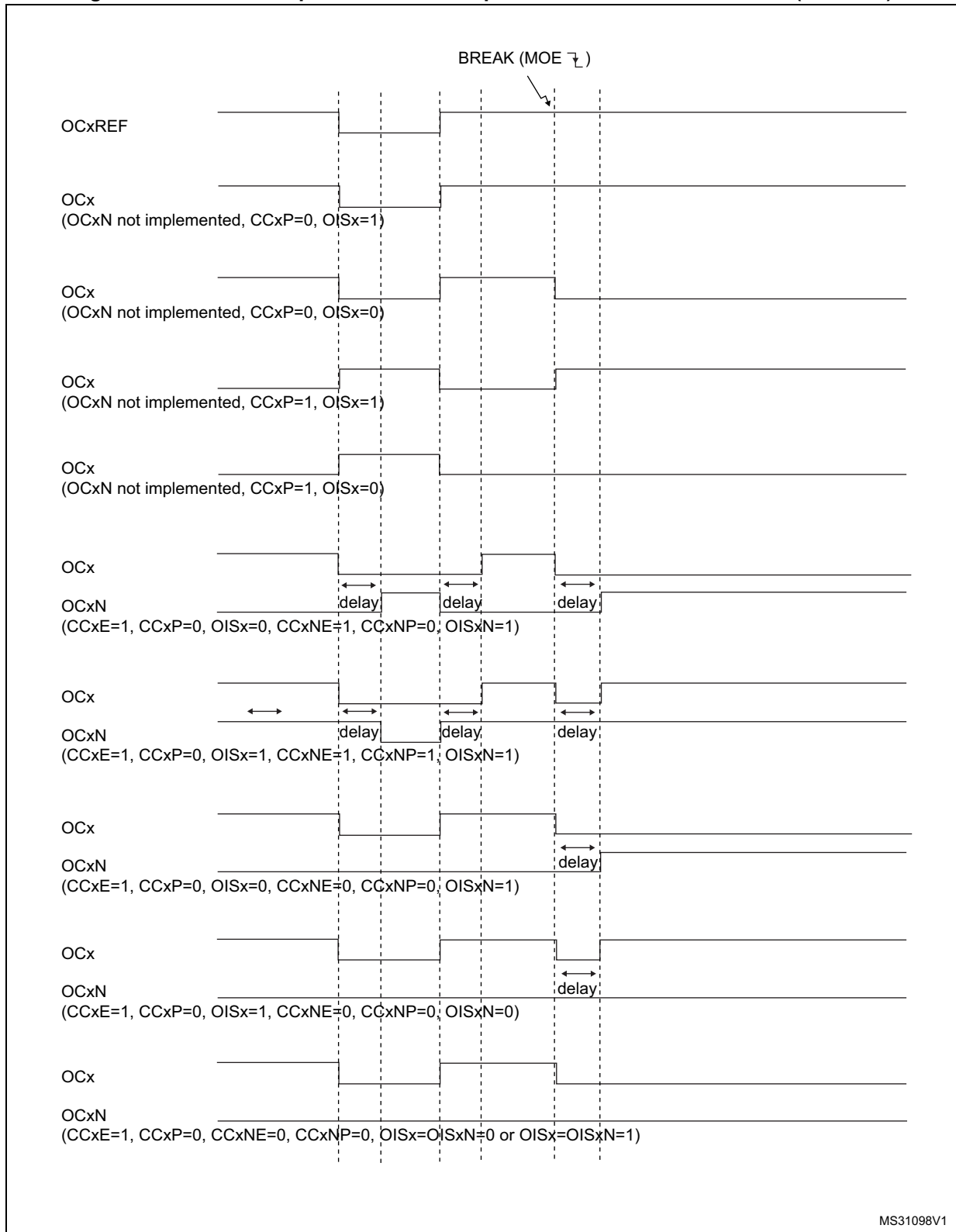
*Note:* If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSS1 value.  
If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the 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 frozen (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx\_BDTR register. Refer to [Section 21.4.20: TIM1 break and dead-time register \(TIM1\\_BDTR\)](#). The LOCK bits can be written only once after an MCU reset.

[Figure 145](#) shows an example of behavior of the outputs in response to a break.

Figure 145. Various output behavior in response to a break event on BRK (OSS1 = 1)





The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in [Table 116](#).

Note: *BRK2 must only be used with OSSR = OSS1 = 1.*

**Table 116. 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 OSS1 = 0 (control taken over by GPIO logic)</li> </ul>	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

[Figure 146](#) 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 146. PWM output state following BRK and BRK2 pins assertion (OSS1=1)**

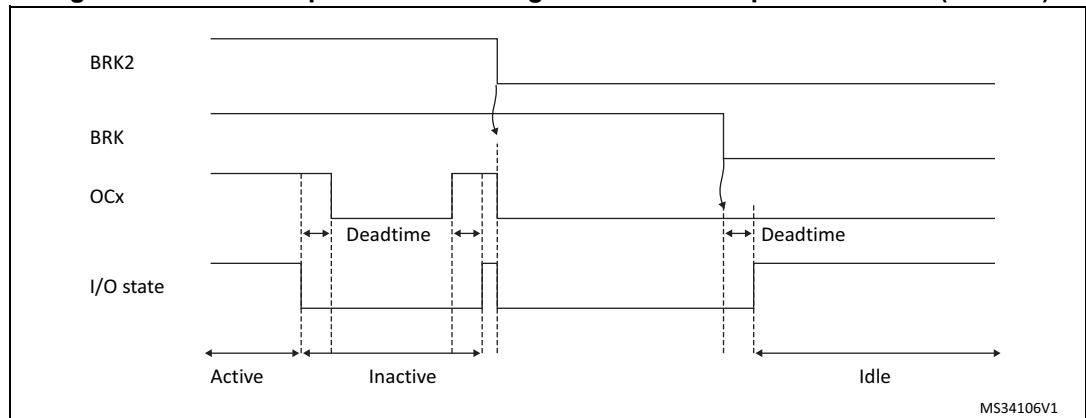
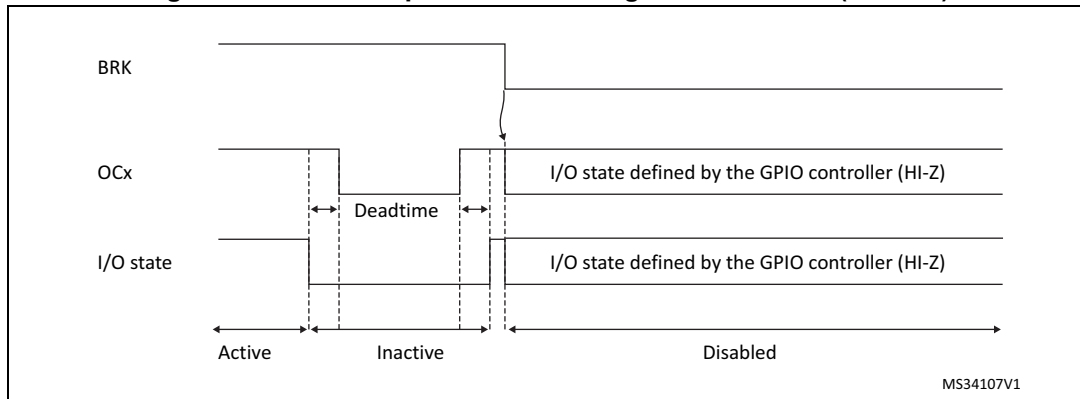


Figure 147. PWM output state following BRK assertion (OSS1=0)



### 21.3.17 Bidirectional break inputs

The TIM1 are featuring bidirectional break I/Os, as represented on [Figure 148](#).

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, 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 117](#))

Table 117. Break protection disarming conditions

MOE	BKDIR (BK2DIR)	BKDSRM (BK2DSRM)	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**

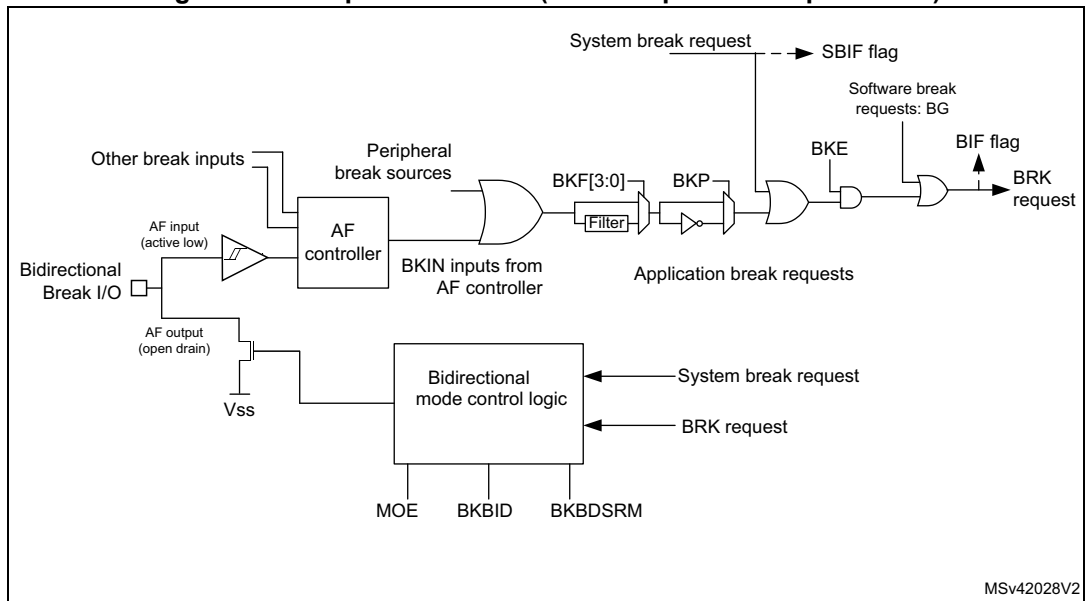
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (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 148. Output redirection (BRK2 request not represented)



### 21.3.18 Clearing the OCxREF signal on an external event

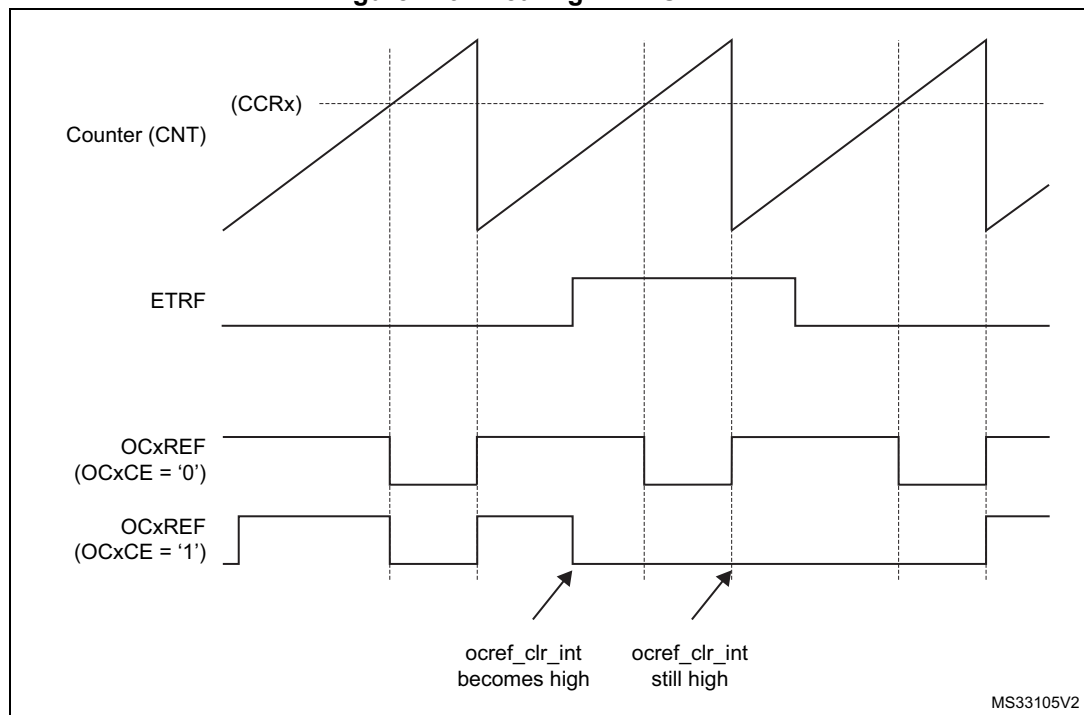
The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref\_clr\_int input (OCxCE enable bit in the corresponding TIMx\_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be 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 149 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 149. 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.

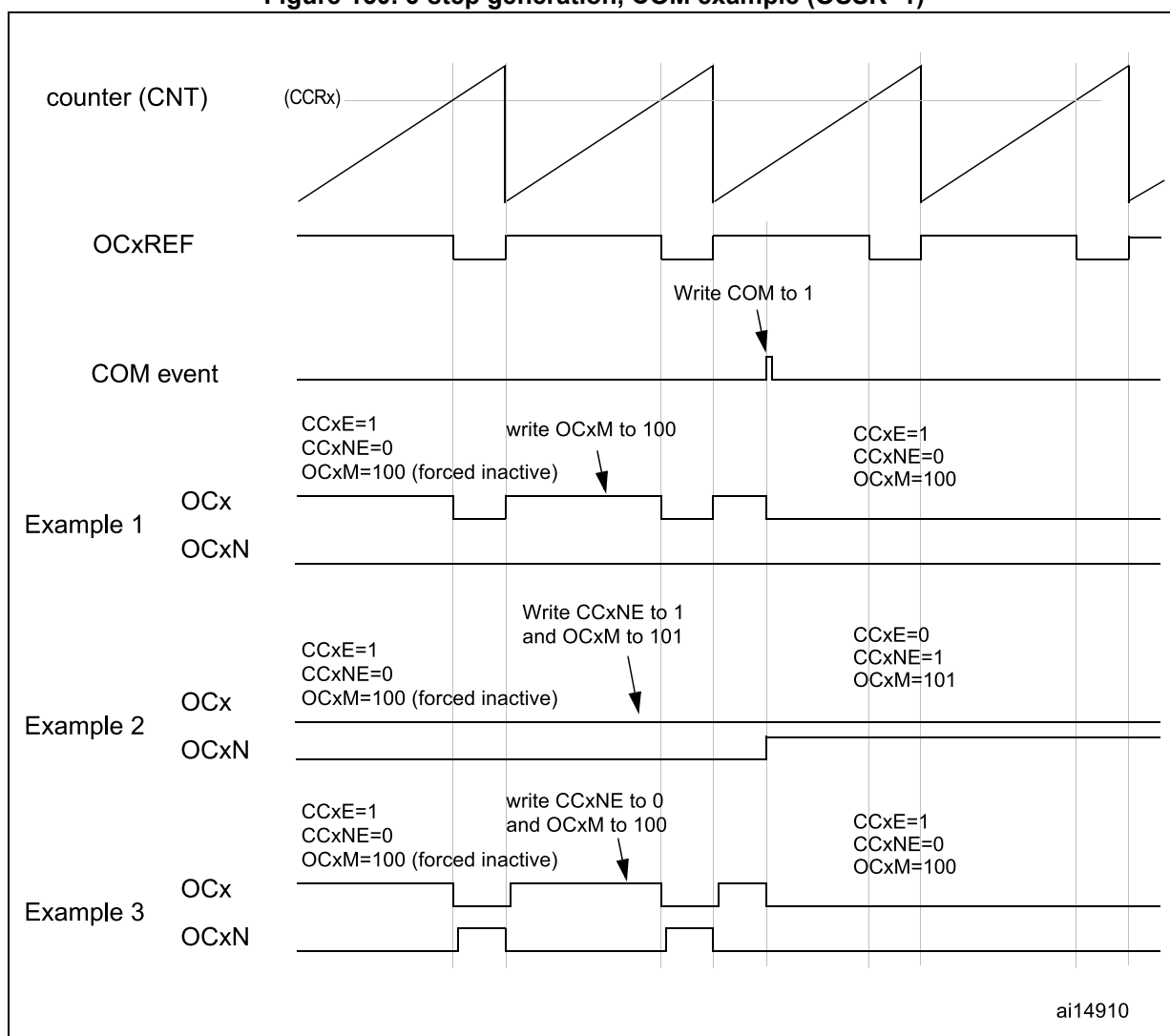
### 21.3.19 6-step PWM generation

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 150](#) describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

**Figure 150. 6-step generation, COM example (OSSR=1)**



### 21.3.20 One-pulse mode

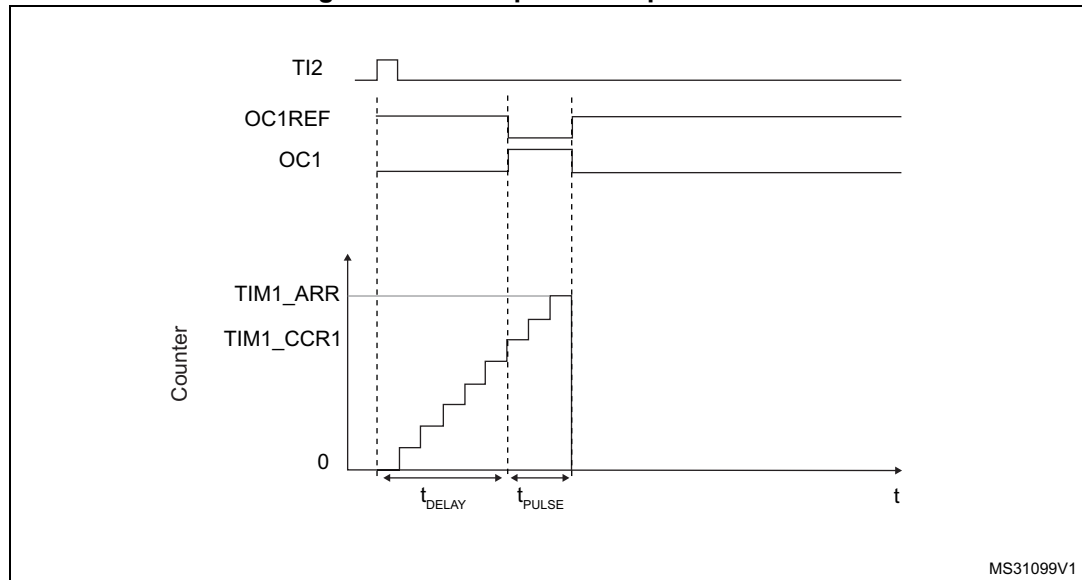
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 151. Example of one pulse mode.**



For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx\_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx\_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx\_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx\_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case one has to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx\_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY min}}$  we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx\_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 21.3.21 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 21.3.20](#):

- 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

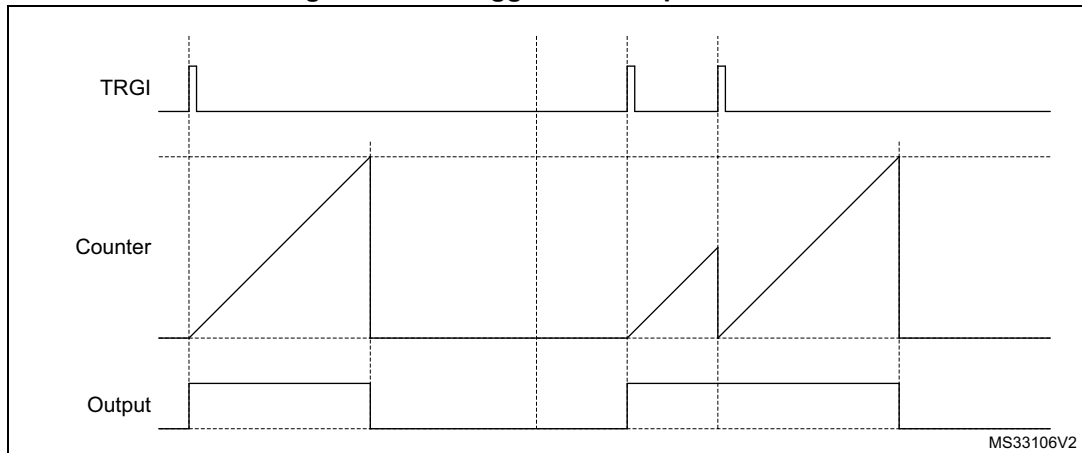
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx\_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for retriggerable OPM mode 1 or 2.

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 152. Retriggerable one pulse mode



### 21.3.22 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 118](#). 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 118. Counting direction versus encoder signals**

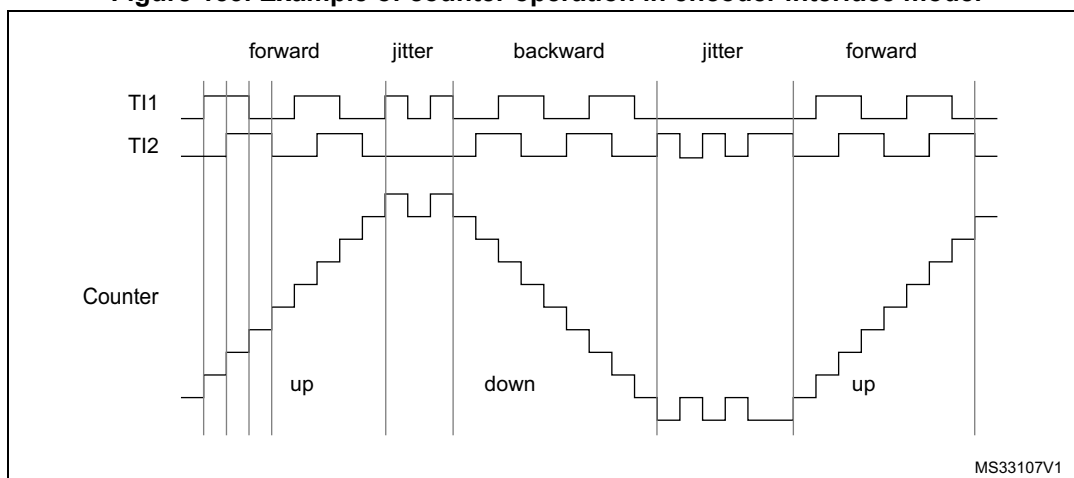
Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting 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 153* 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. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx\_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx\_CCMR1 register, TI2FP2 mapped on TI2)
- CC1P='0' and CC1NP='0' (TIMx\_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx\_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx\_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx\_CR1 register, Counter enabled).

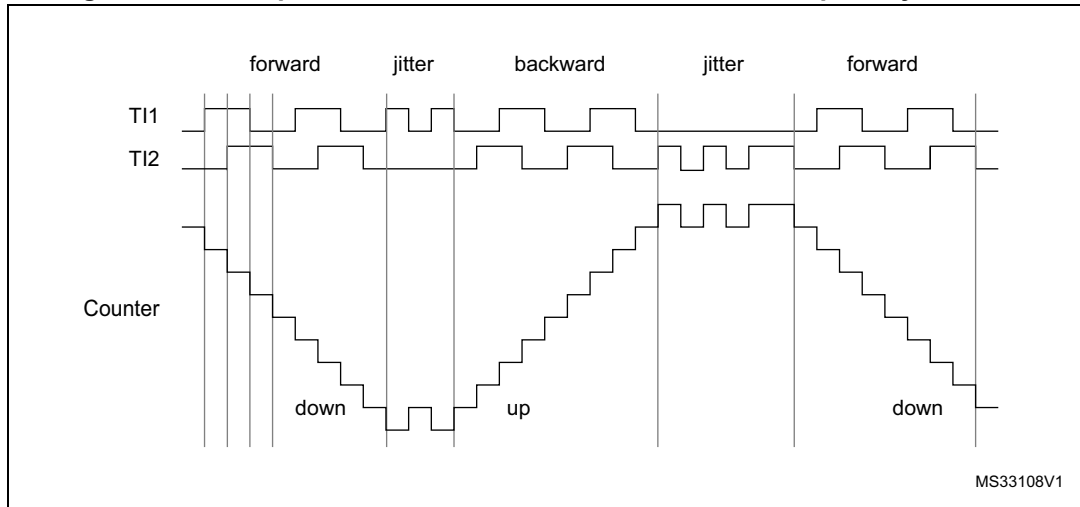
**Figure 153. Example of counter operation in encoder interface mode.**



MS33107V1

Figure 154 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

**Figure 154. 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).

### 21.3.23 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).

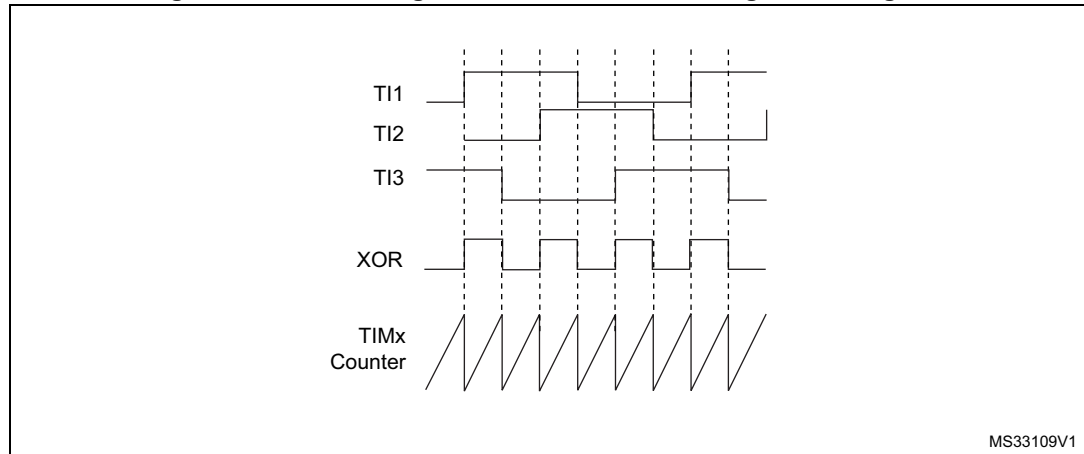
There is no latency between the UIF and UIFCPY flags assertion.

### 21.3.24 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 155](#) below.

**Figure 155. Measuring time interval between edges on 3 signals**



MS33109V1

### 21.3.25 Interfacing with Hall sensors

This is done using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4<sup>(a)</sup>) referred to as “interfacing timer” in [Figure 156](#). 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 129: Capture/compare channel \(example: channel 1 input stage\) on page 538](#)). 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.

a. TIM4 is available on STM32G0B1xx and STM32G0C1xx salestypes only.

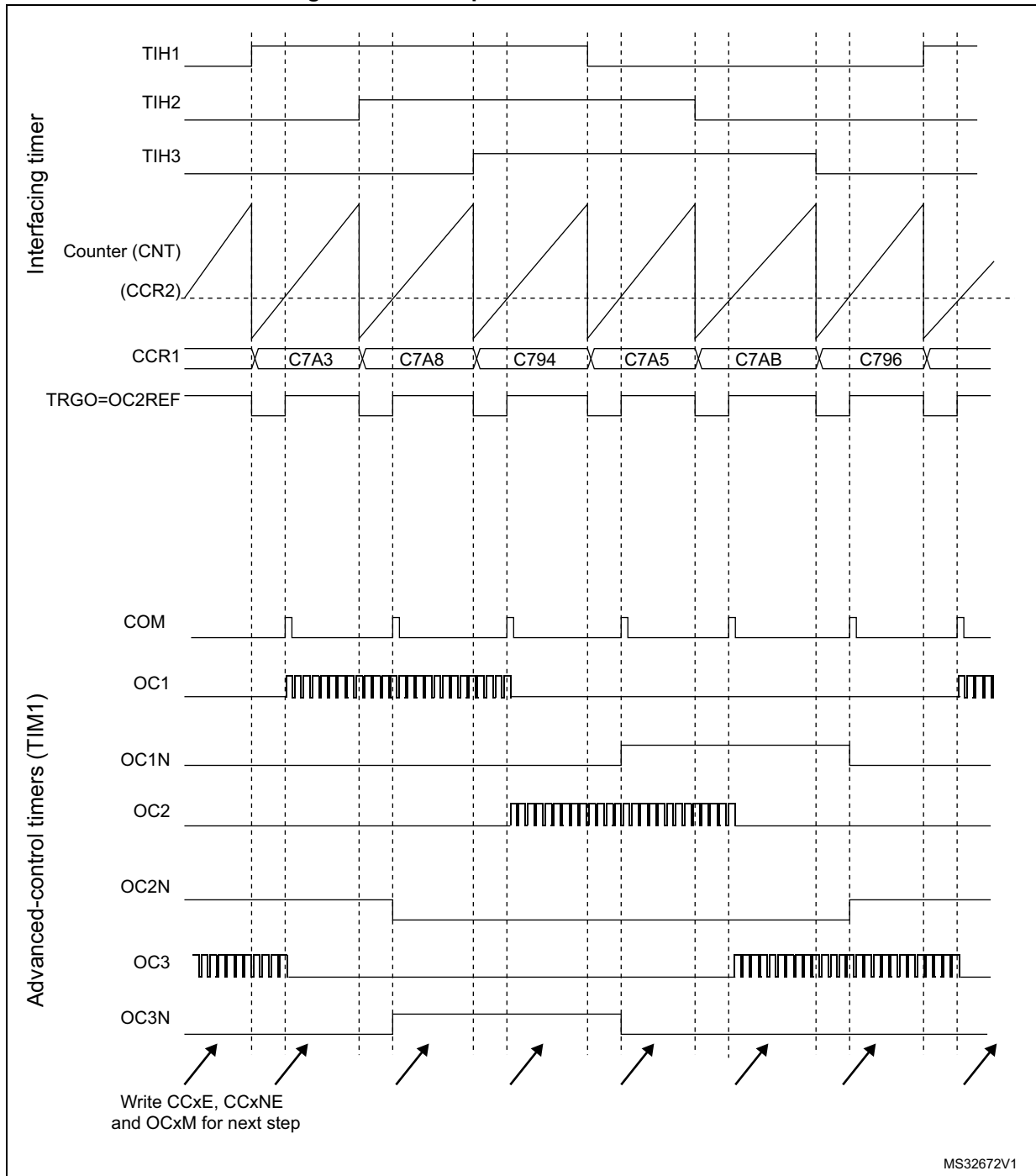
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- 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 '11'. 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 156](#) describes this example.

Figure 156. Example of Hall sensor interface



### 21.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to [Section 22.3.19: 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.

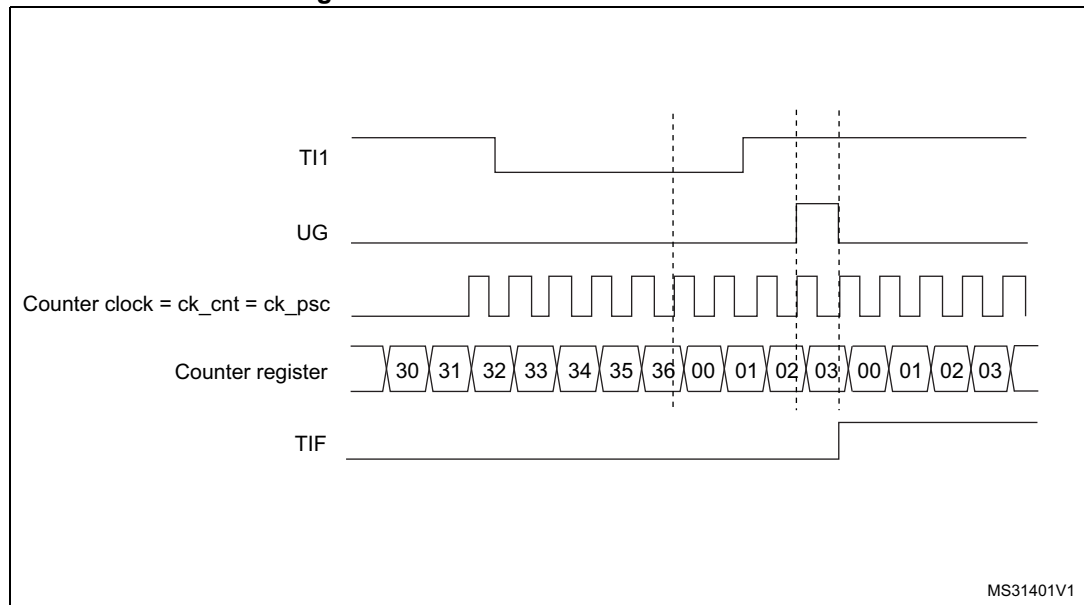
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- 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 157. Control circuit in reset mode**



**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

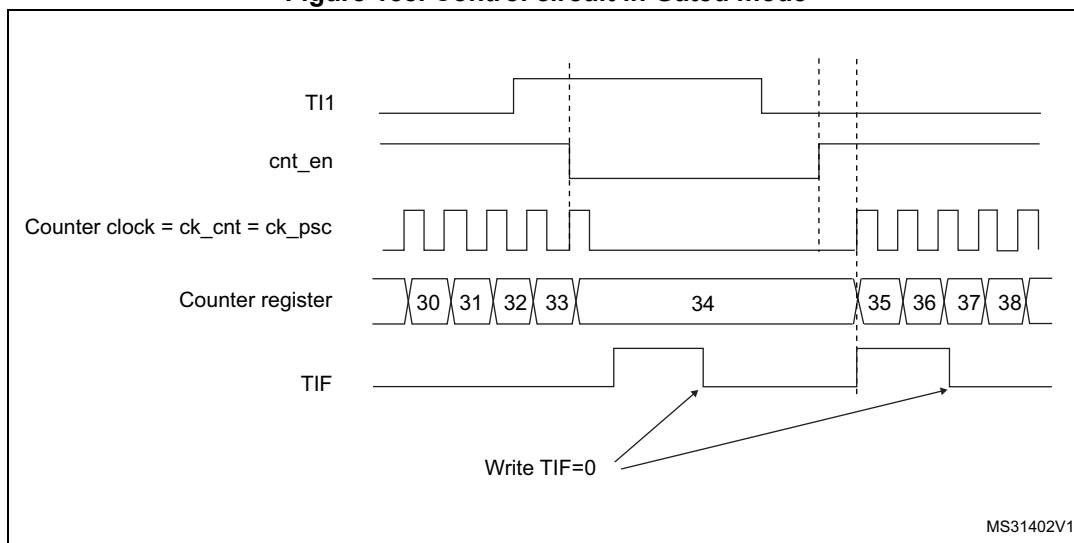
In the following example, the upcounter counts only when TI1 input is low:

- 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 158. 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

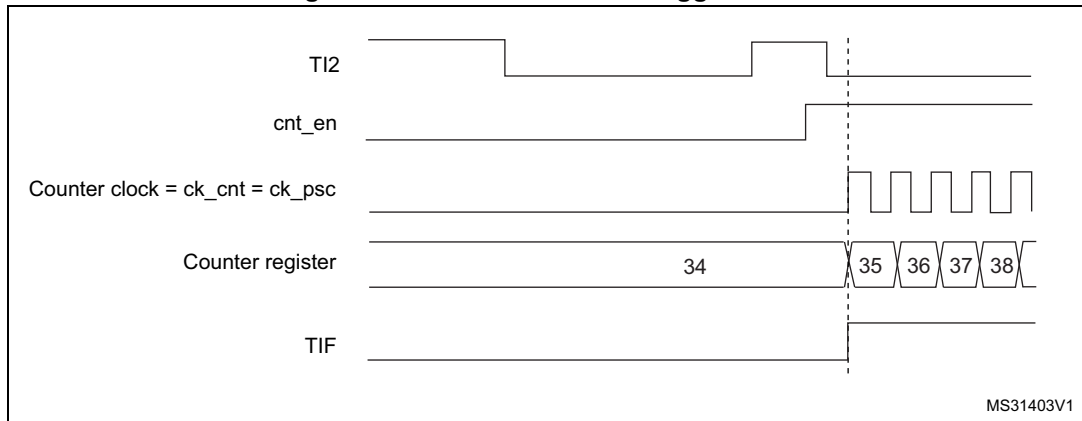
register. Write CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).

- 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 159. 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.



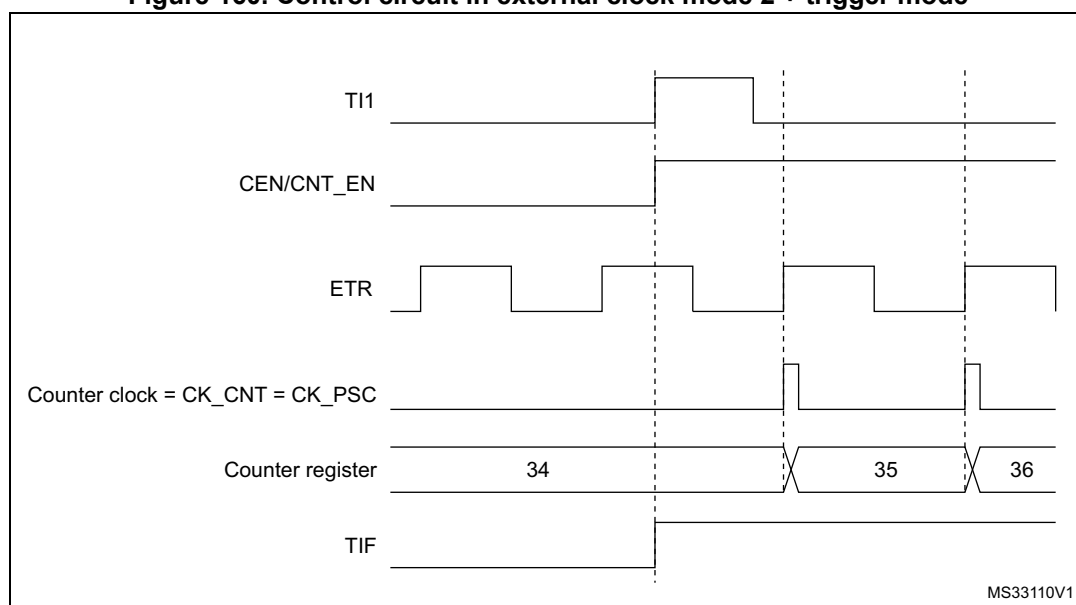
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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 TI1:
  - IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - CC1S = 01 in TIMx\_CCMR1 register to select only the input capture source
  - CC1P = 0 and CC1NP = 0 in TIMx\_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=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 160. 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.

### 21.3.27 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 140 on page 550](#).

*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.

### 21.3.28 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.

### 21.3.29 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).

## 21.4 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).

### 21.4.1 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.*

### 21.4.2 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.*

### 21.4.3 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]
										r/w	r/w				r/w
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]		
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:22 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 119: TIM1 internal trigger connection on page 582](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source.

- 0: OCREF\_CLR\_INT is connected to COMP1 or COMP2 output depending on TIM1\_OR1.OCREF\_CLR
- 1: OCREF\_CLR\_INT is connected to ETRF

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (refer to ETP bit in TIMx\_SMCR for tim\_etr\_in and CCxP/CCxNP bits in TIMx\_CCER register for tim\_ti1fp1 and tim\_ti2fp2).

- 0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.
- 0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
- 0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
- 0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
- 0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
- 0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
- 0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
- 0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
- 1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
- Codes above 1000: Reserved.

*Note: The gated mode must not be used if TI1F\_ED is selected as the trigger input (TS=00100). Indeed, TI1F\_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.*

*Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.*

**Table 119. TIM1 internal trigger connection**

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM1	TIM15	TIM2	TIM3	TIM17 OC1

### 21.4.4 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

### 21.4.5 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.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	B1F	TIF	COMIF	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	rc_w0	rc_w0	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 **SBIF**: 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 21.4.3: TIM1 slave mode control register \(TIM1\\_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 21.4.6 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.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

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).

### 21.4.7 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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:12 **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).*

## 21.4.8 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. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

### 21.4.9 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).*

### 21.4.10 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).*

### 21.4.11 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
										r/w	r/w			r/w	r/w
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
r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 21 **CC6P**: Capture/Compare 6 output polarity  
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable  
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity  
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable  
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity  
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity  
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable  
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity  
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable  
Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity  
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable  
Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity  
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable  
Refer to CC1NE description

- Bit 5 **CC2P**: Capture/Compare 2 output polarity  
Refer to CC1P description
- Bit 4 **CC2E**: Capture/Compare 2 output enable  
Refer to CC1E description
- Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity  
**CC1 channel configured as output:**  
 0: OC1N active high.  
 1: OC1N active low.  
**CC1 channel configured as input:**  
 This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.  
*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (channel configured as output).*  
*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.*
- Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable  
 0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
 1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
*On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.*
- Bit 1 **CC1P**: Capture/Compare 1 output polarity  
 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)  
 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)  
 When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.  
 CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).  
 CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).  
 CC1NP=1, CC1P=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, OSSI, OSSI, OSSI1, OSSI1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 120](#) 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 120. 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	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
			0	0		
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered).	
			1	0		
			1	1	Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). <b>Note:</b> BRK2 can only be used if OSSI = OSSR = 1.	

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

*Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.*



### 21.4.12 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

### 21.4.13 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 21.4.14 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 21.3.1: Time-base unit on page 518](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 21.4.15 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.

### 21.4.16 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.

### 21.4.17 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.

### 21.4.18 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.

### 21.4.19 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **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.

### 21.4.20 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	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
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]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

*Note:* As the bits BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx\_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional  
Refer to BKBID description

Bit 28 **BKBID**: Break Bidirectional

- 0: Break input BRK in input mode
- 1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 27 **BK2DSRM**: Break2 Disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break Disarm

- 0: Break input BRK is armed
- 1: Break input BRK is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 25 **BK2P**: Break 2 polarity

- 0: Break input BRK2 is active low
- 1: Break input BRK2 is active high

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 24 **BK2E**: Break 2 enable

- 0: Break input BRK2 disabled
- 1: Break input BRK2 enabled

*Note: The BRK2 must only be used with OSSR = OSSR = 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 OSSR 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 21.4.11: 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\_ach and BRK sources, as per [Figure 144: 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 21.4.11: TIM1 capture/compare enable register \(TIM1\\_CCER\)](#)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details ([Section 21.4.11: 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, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits in TIMx\_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx\_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx\_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx\_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x 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).*

### 21.4.21 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



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,

...

### 21.4.22 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).

### 21.4.23 TIM1 option register 1 (TIM1\_OR1)

Address offset: 0x50

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.	Res.	Res.	OCREF_CLR [1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1:0 **OCREF\_CLR[1:0]**: Ocref\_clr source selection

This bit selects the ocref\_clr input source.

00: COMP1 output is connected to the OCREF\_CLR input

01: COMP2 output is connected to the OCREF\_CLR input

10: COMP3 output is connected to the OCREF\_CLR input

11: Reserved

Note: COMP3 is available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 21.4.24 TIM1 capture/compare mode register 3 (TIM1\_CCMR3)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

**Output compare mode:**

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.

### 21.4.25 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r/w	r/w	r/w													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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.

### 21.4.26 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.

### 21.4.27 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.	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.	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
rw	rw		rw	rw	rw	rw						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

0001: COMP1 output

0010: COMP2 output

0011: ADC1 AWD1

0100: ADC1 AWD2

0101: ADC1 AWD3

0110: COMP3 output (available on STM32G0B1xx and STM32G0C1xx salestypes only)

Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **BKCOMP3P**: BRK COMP3 input polarity

This bit selects the COMP3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP3 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP3 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).

Note: This bit is available on STM32G0B1xx and STM32G0C1xx salestypes only, reserved otherwise.

Bit 11 **BKCOMP2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP2 input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 10 **BKCMP1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: COMP1 input polarity is inverted (active high if BKP=0, active low if BKP=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BKCMP3E**: BRK COMP3 enable

This bit enables the COMP3 for the timer's BRK input. COMP3 output is 'ORed' with the other BRK sources.

0: COMP3 input disabled

1: COMP3 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: This bit is available on STM32G0B1xx and STM32G0C1xx salestypes only), reserved otherwise.*

Bit 2 **BKCMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Note:** Refer to [Figure 123: TIM1 ETR input circuitry](#) and to [Figure 144: Break and Break2 circuitry overview](#).

### 21.4.28 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.	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.	BK2 CMP3 P	BK2 CMP2 P	BK2 CMP1 P	BK2 INP	Res.	Res.	Res.	Res.	Res.	BK2 CMP3E	BK2 CMP2E	BK2 CMP1E	BK2INE
			rw	rw	rw	rw						rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BK2CMP3P**: BRK2 COMP3 input polarity

This bit selects the COMP3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP3 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP3 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).*

*Note: This bit is available on STM32G0B1xx and STM32G0C1xx salestypes only), reserved otherwise.*

Bit 11 **BK2CMP2P**: BRK2 COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BK2CMP1P**: BRK2 COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: COMP1 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: COMP1 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:4 Reserved, must be kept at reset value.

- Bit 3 **BK2CMP3E**: BRK2 COMP3 enable  
 This bit enables the COMP3 for the timer’s BRK2 input. COMP3 output is ‘ORed’ with the other BRK2 sources.  
 0: COMP3 input disabled  
 1: COMP3 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: This bit is available on STM32G0B1xx and STM32G0C1xx salestypes only), reserved otherwise.*
- Bit 2 **BK2CMP2E**: BRK2 COMP2 enable  
 This bit enables the COMP2 for the timer’s BRK2 input. COMP2 output is ‘ORed’ with the other BRK2 sources.  
 0: COMP2 input disabled  
 1: COMP2 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Bit 1 **BK2CMP1E**: BRK2 COMP1 enable  
 This bit enables the COMP1 for the timer’s BRK2 input. COMP1 output is ‘ORed’ with the other BRK2 sources.  
 0: COMP1 input disabled  
 1: COMP1 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*
- Bit 0 **BK2INE**: BRK2 BKIN input enable  
 This bit enables the BKIN2 alternate function input for the timer’s BRK2 input. BKIN2 input is ‘ORed’ with the other BRK2 sources.  
 0: BKIN2 input disabled  
 1: BKIN2 input enabled  
*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Note: Refer to [Figure 144: Break and Break2 circuitry overview](#).

### 21.4.29 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

0001: COMP3 output (available on STM32G0B1xx and STM32G0C1xx salestypes only)

Others: Reserved

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

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM1\_CH2 input

0001: COMP2 output

Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM1\_CH1 input

0001: COMP1 output

Others: Reserved

### 21.4.30 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 121. TIM1 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	TIM1_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UJFREMAP	Res.	CKD [1:0]	ARPE	Res.	CMS [1:0]	DIR	OPM	URS	UDIS	CEN	
	Reset value																					0		0	0	0	0	0	0	0	0	0	
0x04	TIM1_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]			Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	Res.	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]		CCDS	CCUS	Res.	CCPC	
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	TIM1_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]	Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP S [1:0]		ETF[3:0]			MSM	TS[2:0]		OCCS	SMS[2:0]					
	Reset value											0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	TIM1_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x10	TIM1_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	
0x14	TIM1_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG	
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	TIM1_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]		OC2PE	OC2FE	CC2 S [1:0]	OC1CE	OC1M [2:0]		OC1PE	OC1FE	CC1 S [1:0]				
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	TIM1_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]			IC2 PSC [1:0]	CC2 S [1:0]	IC1F[3:0]			IC1 PSC [1:0]	CC1 S [1:0]						
Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0			
0x1C	TIM1_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]		OC4PE	OC4FE	CC4 S [1:0]	OC3CE	OC3M [2:0]		OC3PE	OC3FE	CC3 S [1:0]				
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	TIM1_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]			IC4 PSC [1:0]	CC4 S [1:0]	IC3F[3:0]			IC3 PSC [1:0]	CC3 S [1:0]						
Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x20	TIM1_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 121. TIM1 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	TIM1_CNT	UIFCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIM1_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIM1_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0x30	TIM1_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x34	TIM1_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x38	TIM1_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x3C	TIM1_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x40	TIM1_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x44	TIM1_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]			BKF[3:0]			MOE	AOE	BKP	BKE	OSSR	OSSI	LOK K[1:0]	DT[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	
0x48	TIM1_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]				Res.	Res.	Res.	DBA[4:0]					
	Reset value																				0	0	0	0	0			0	0	0	0	0	
0x4C	TIM1_DMAR	DMAB[31:0]																															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x50	TIM1_OR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCREF_CLR
	Reset value																															0	0
0x54	TIM1_CCMR3 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]		OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]		OC5PE	OC5FE	Res.	Res.	
	Reset value							0										0	0	0	0	0	0			0	0	0	0	0			



Table 121. TIM1 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	
0x58	TIM1_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]																
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x5C	TIM1_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x60	TIM1_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]			Res.	BK2CMP3P(1)	BK2CMP2P	BK2CMP1P	BK2INP	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP3E(1)	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																0	0	0	0		0	0	0	0					0	0	0	1	
0x64	TIM1_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP3P(1)	BK2CMP2P	BK2CMP1P	BK2INP	Res.	Res.	Res.	Res.	Res.	Res.	BK2CMP3E(1)	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value																				0	0	0	0					0	0	0	1		
0x68	TIM1_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]				
	Reset value					0	0	0	0						0	0	0	0					0	0	0	0					0	0	0	0

1. Available on STM32G0B1xx and STM32G0C1xx saletypes only.

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 22 General-purpose timers (TIM2/TIM3/TIM4)

### 22.1 TIM2/TIM3/TIM4 introduction

The general-purpose timers consist of a 16-bit/32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in [Section 22.3.19: Timer synchronization](#).

### 22.2 TIM2/TIM3/TIM4 main features

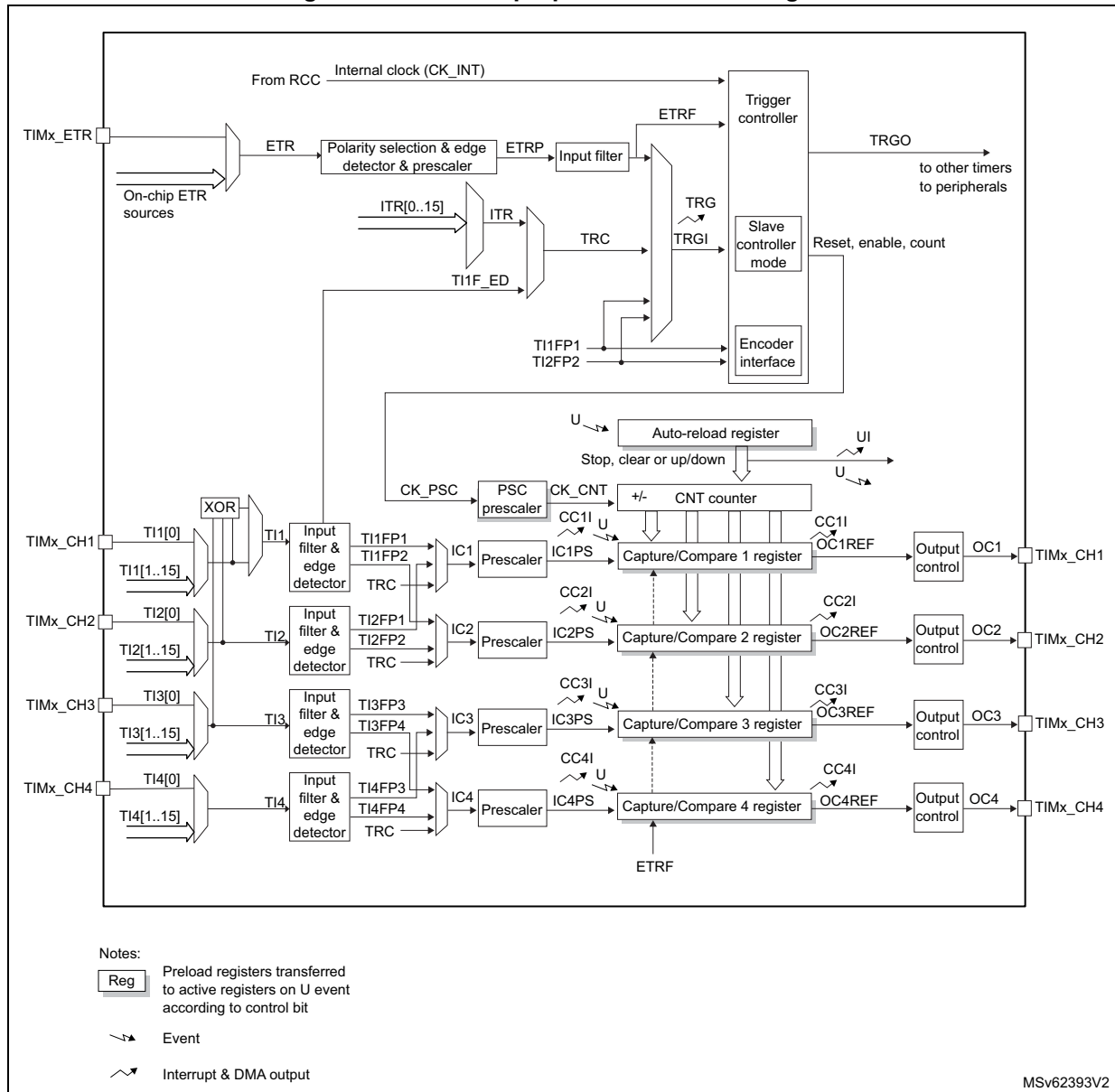
General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4<sup>(a)</sup>) or 32-bit (TIM2) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

---

a. TIM4 is available on the STM32G0B1xx and STM32G0C1xx devices only.

Figure 161. General-purpose timer block diagram



## 22.3 TIM2/TIM3/TIM4 functional description

### 22.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. 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 162](#) and [Figure 163](#) give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 162. Counter timing diagram with prescaler division change from 1 to 2

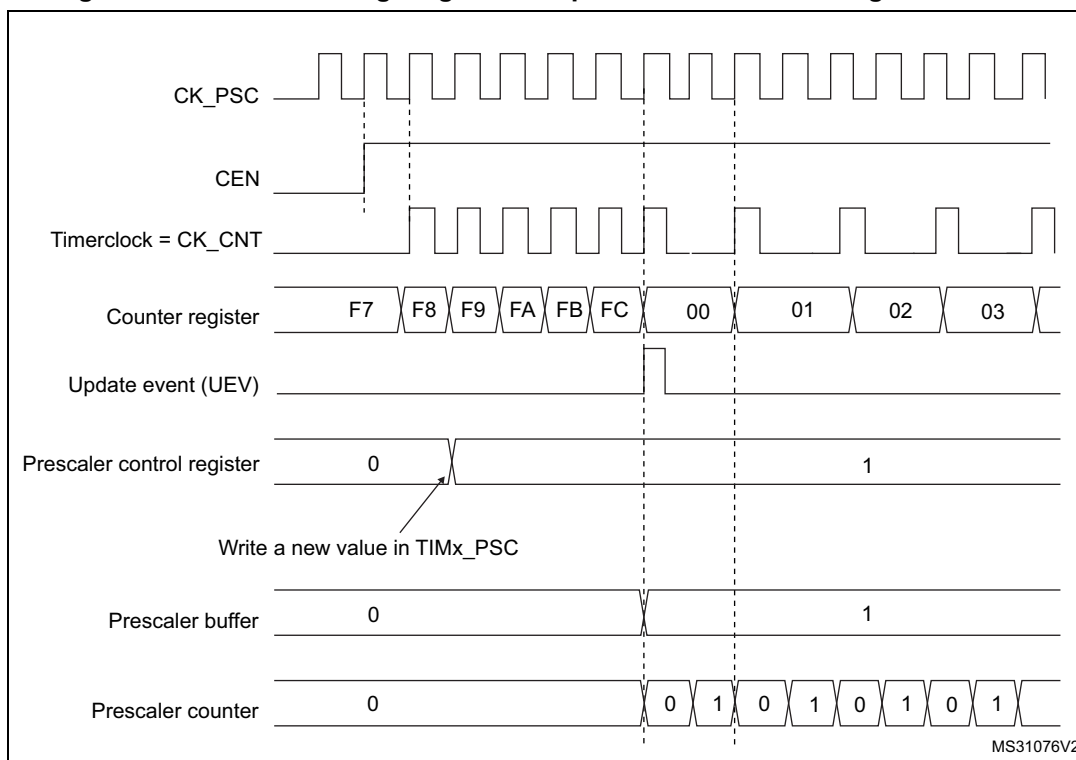
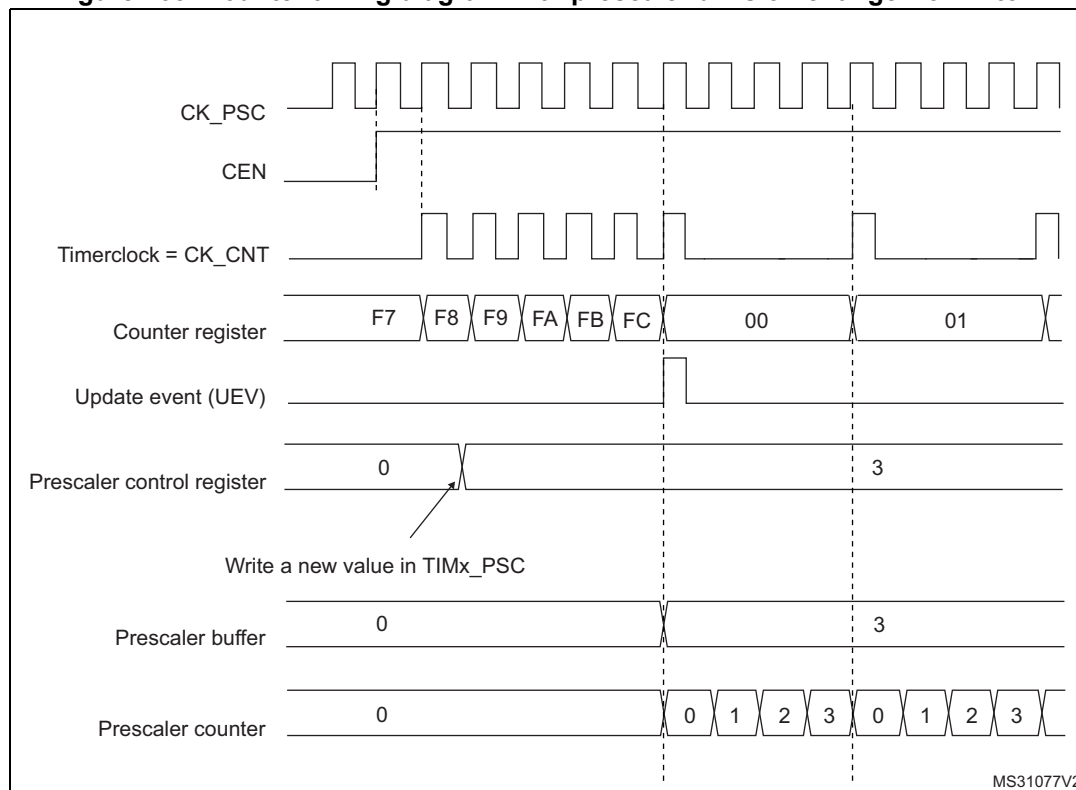


Figure 163. Counter timing diagram with prescaler division change from 1 to 4





### 22.3.2 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 164. Counter timing diagram, internal clock divided by 1**

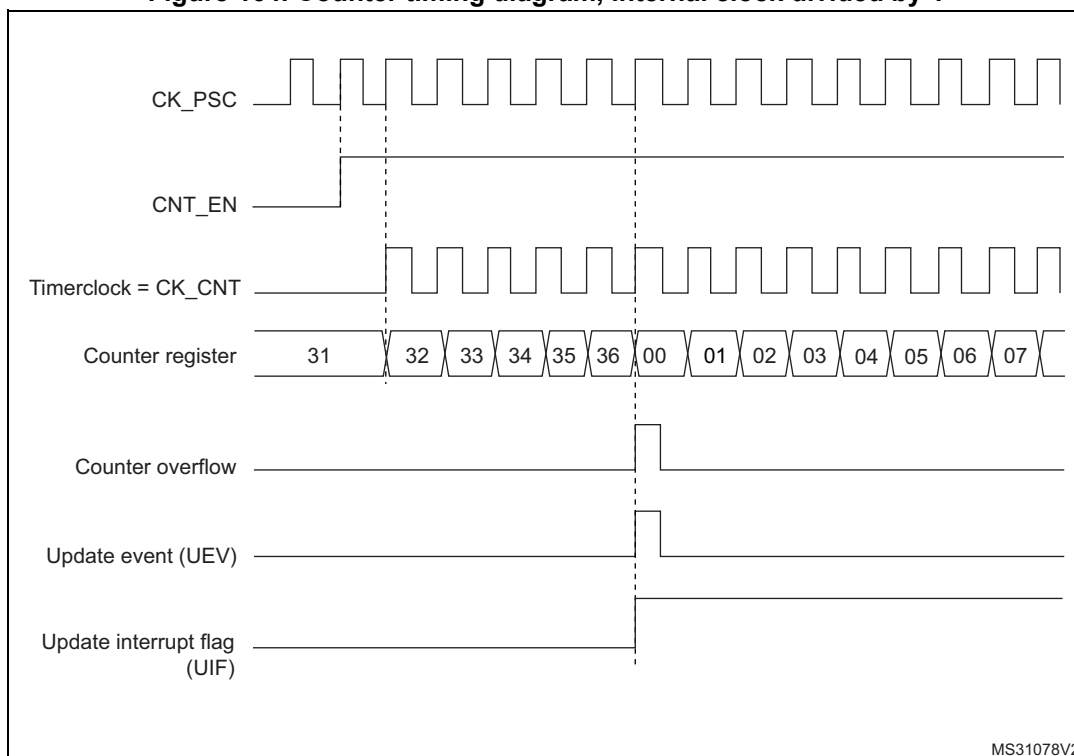
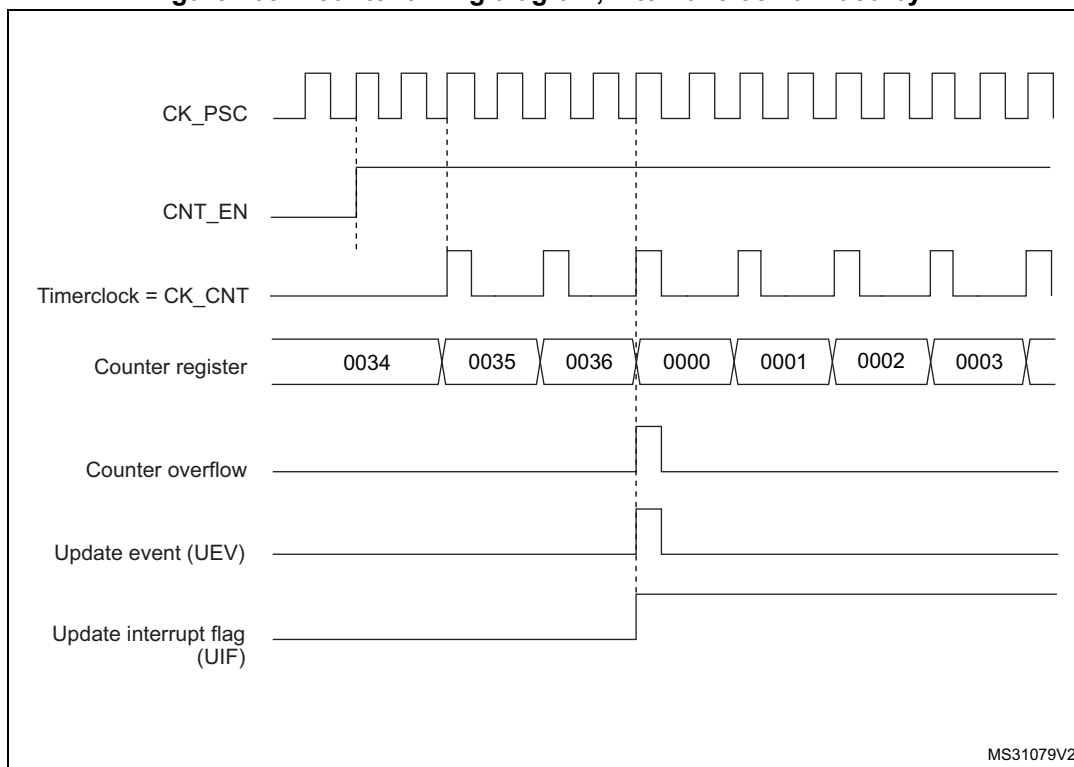
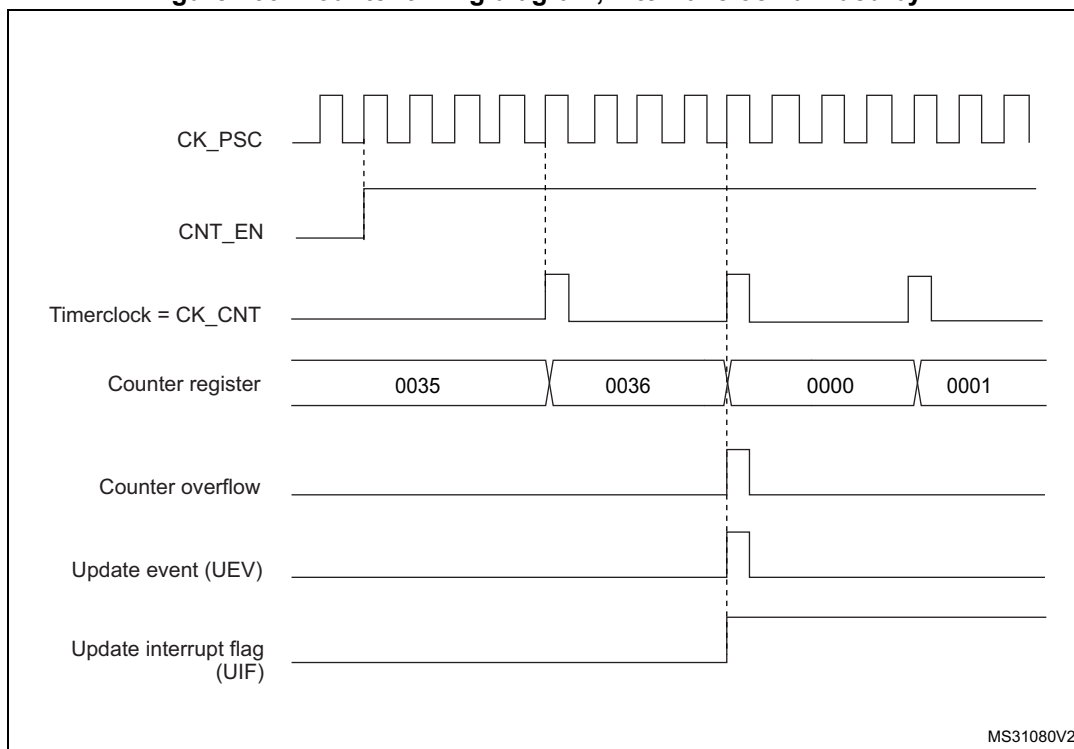


Figure 165. Counter timing diagram, internal clock divided by 2



MS31079V2

Figure 166. Counter timing diagram, internal clock divided by 4



MS31080V2

Figure 167. Counter timing diagram, internal clock divided by N

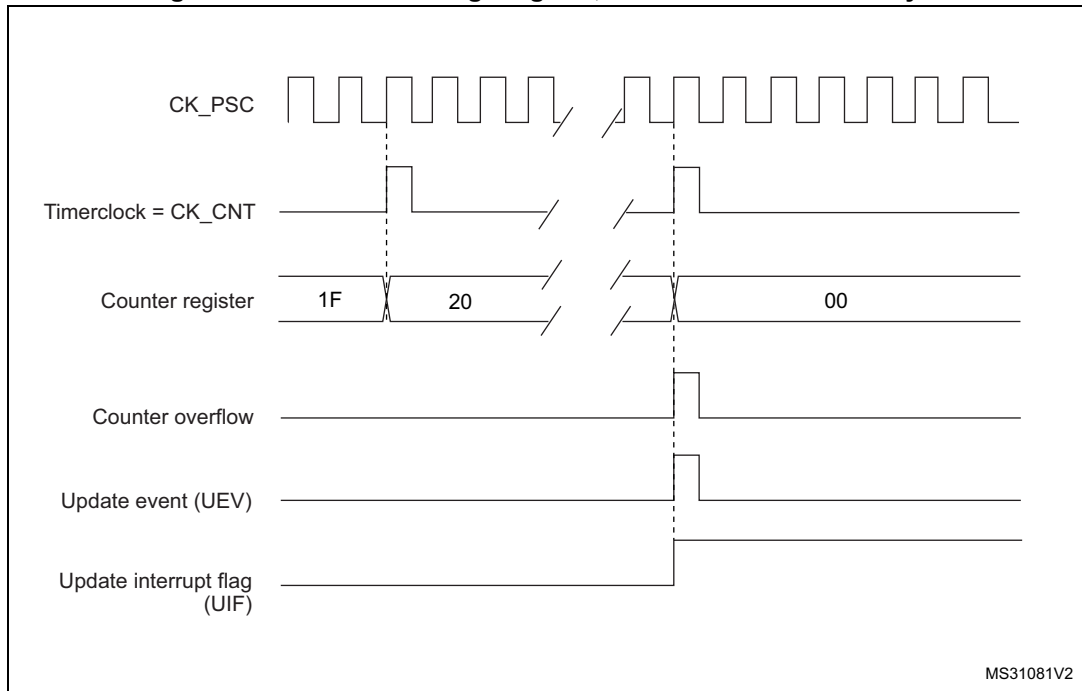
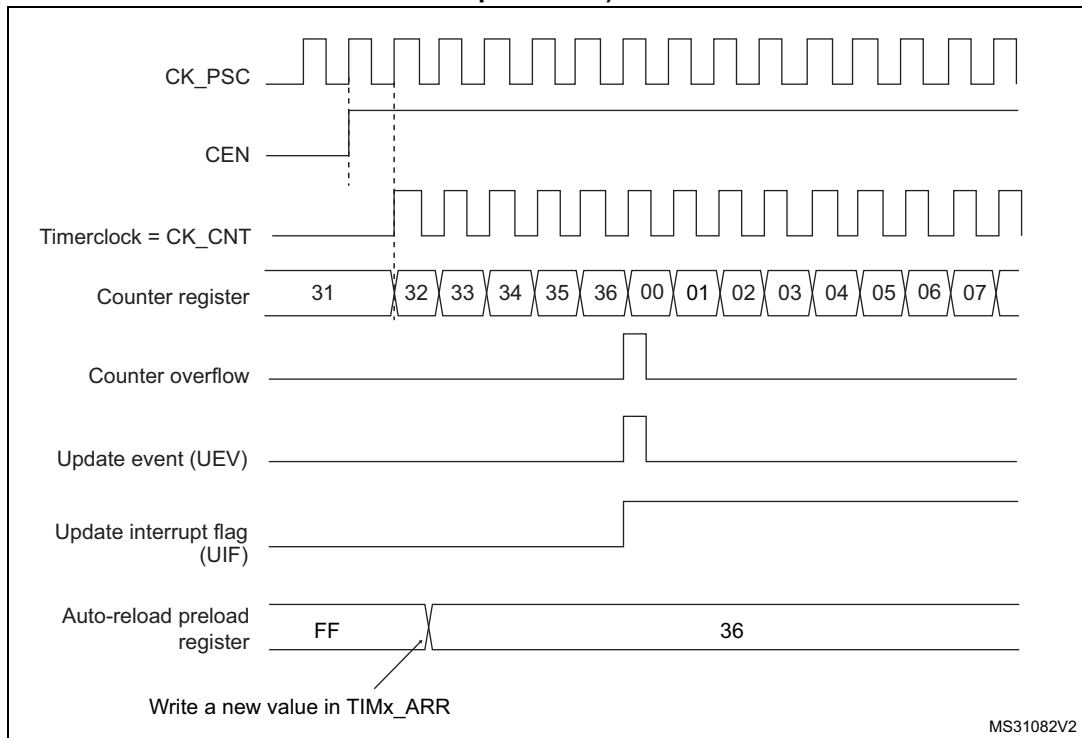
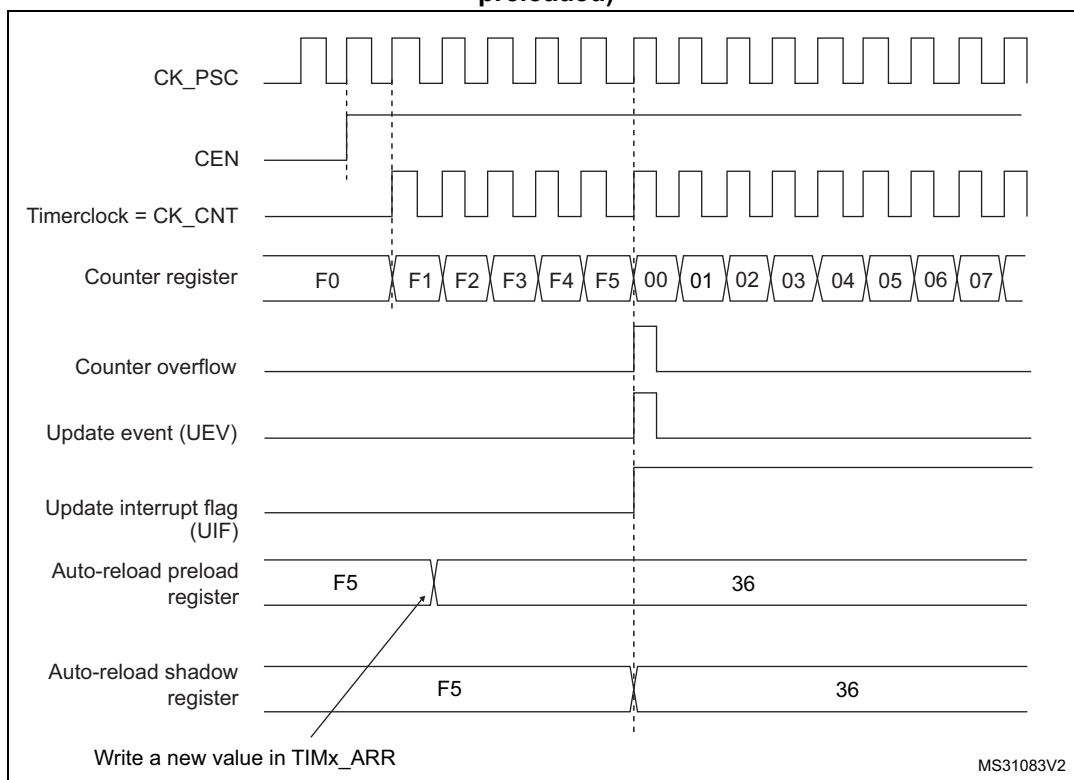


Figure 168. Counter timing diagram, Update event when ARPE=0 (TIMx\_ARR not preloaded)



**Figure 169. Counter timing diagram, Update event when ARPE=1 (TIMx\_ARR preloaded)**



### Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx\_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be 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).

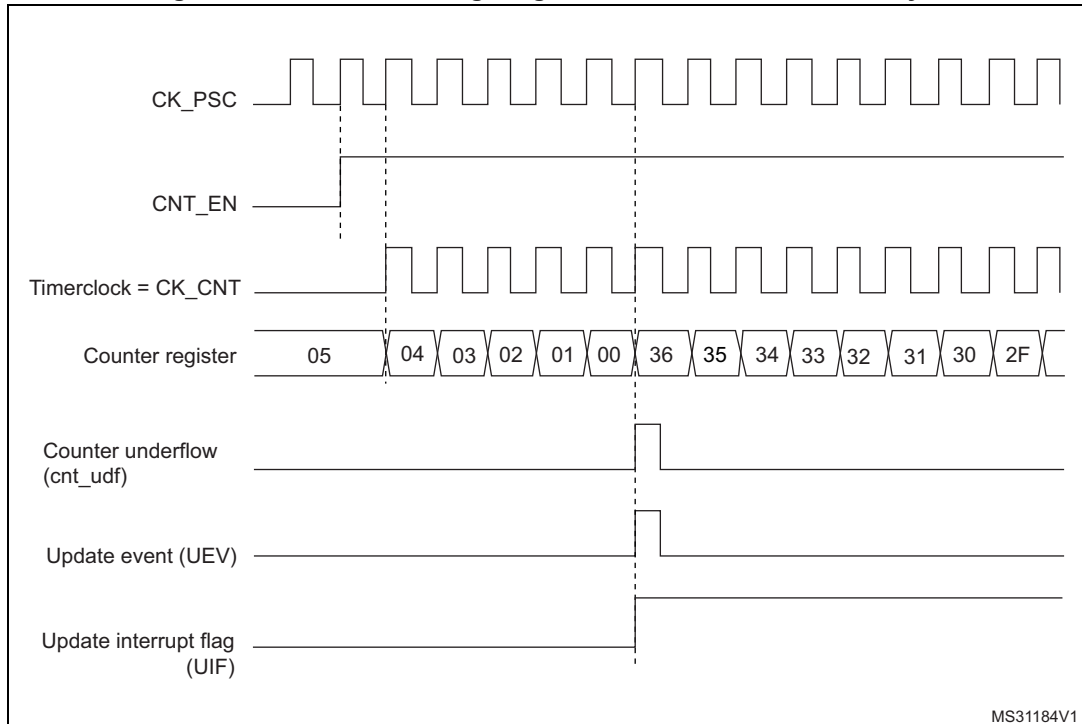
In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 170. Counter timing diagram, internal clock divided by 1**



**Figure 171. Counter timing diagram, internal clock divided by 2**

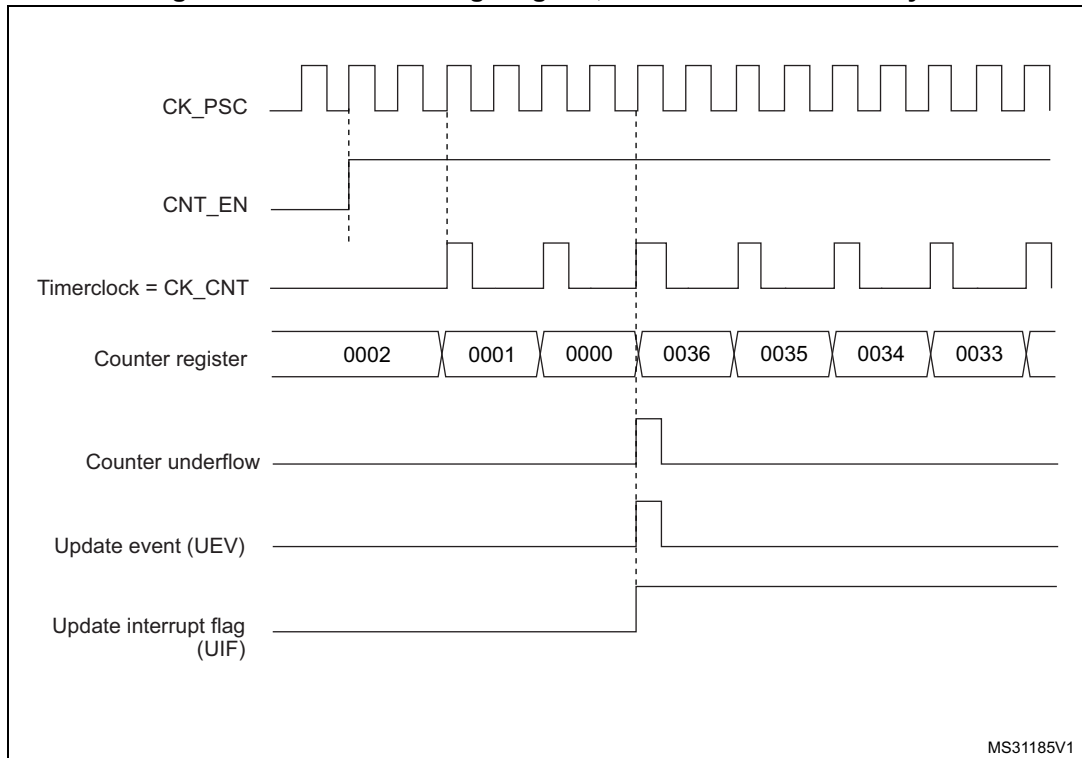


Figure 172. Counter timing diagram, internal clock divided by 4

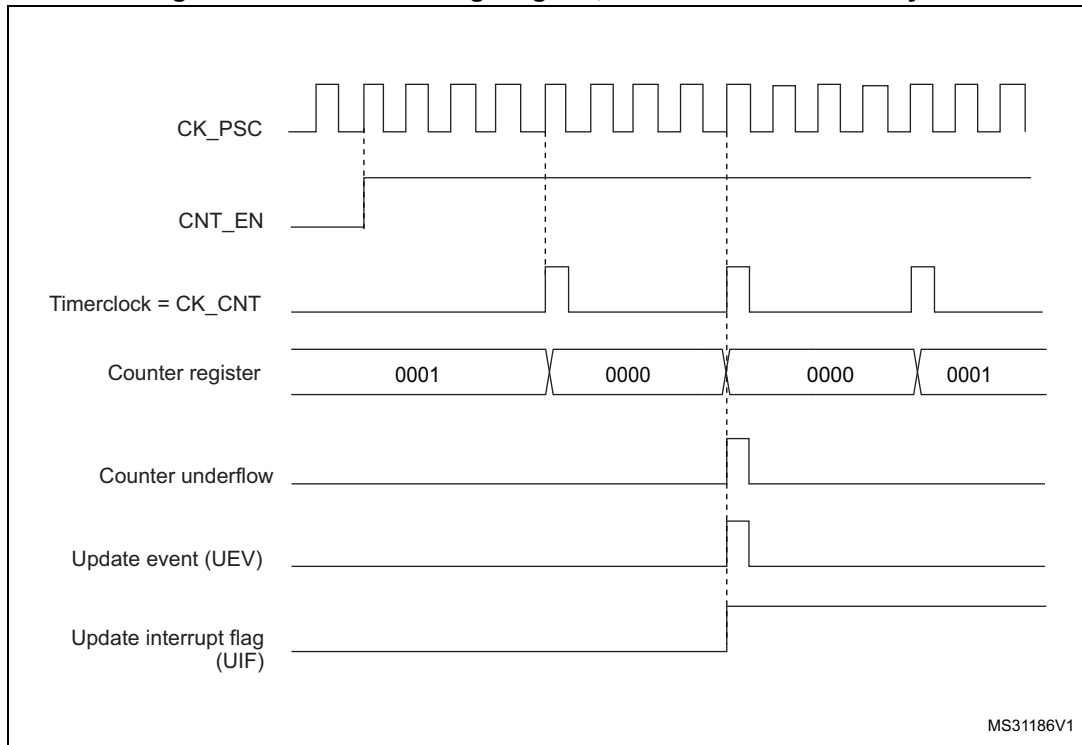


Figure 173. Counter timing diagram, internal clock divided by N

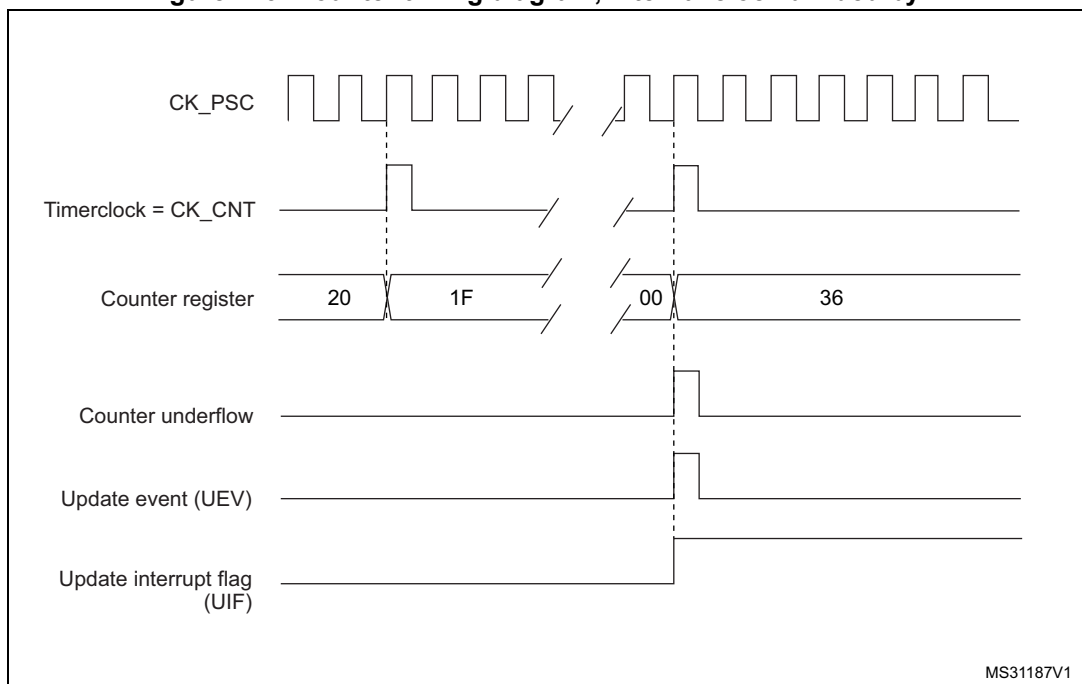
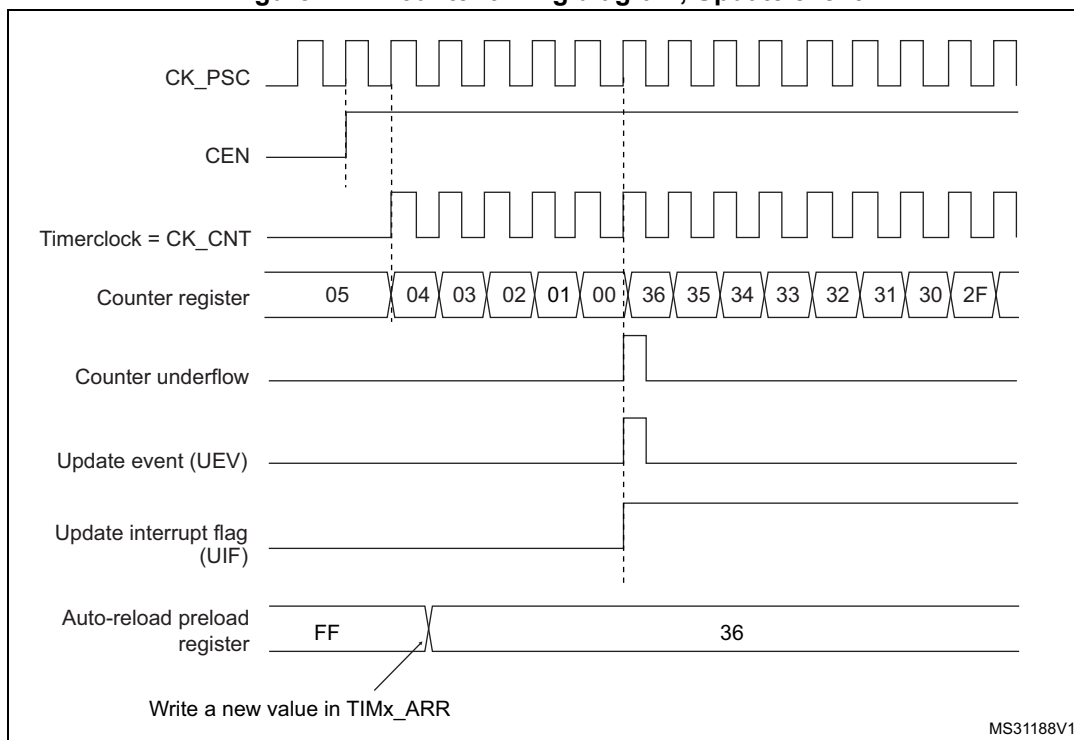


Figure 174. Counter timing diagram, Update event



**Center-aligned mode (up/down counting)**

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx\_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx\_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

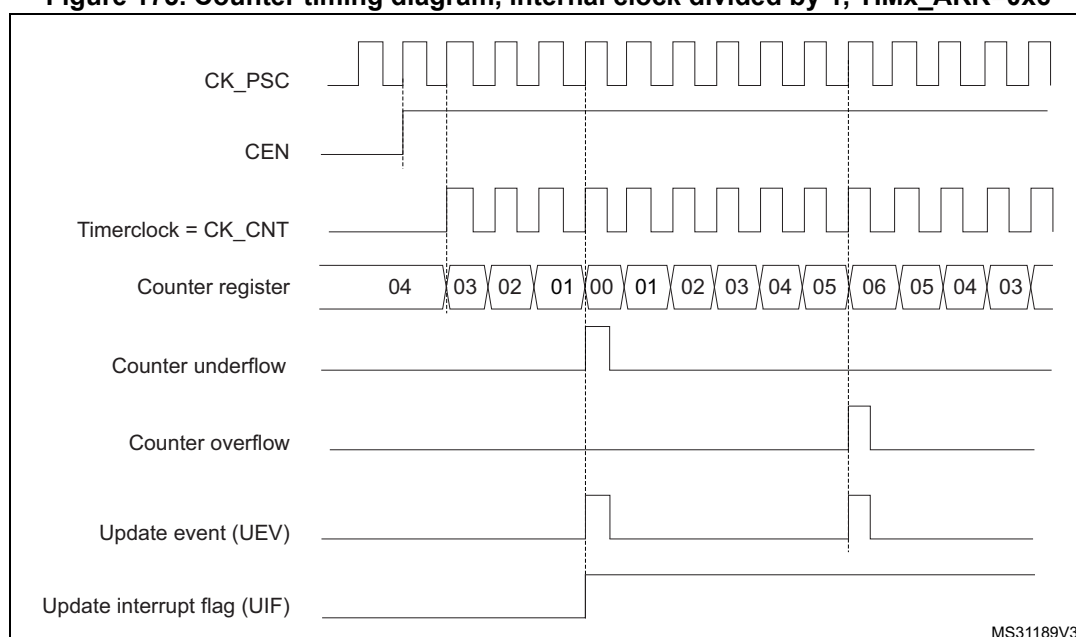
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx\_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

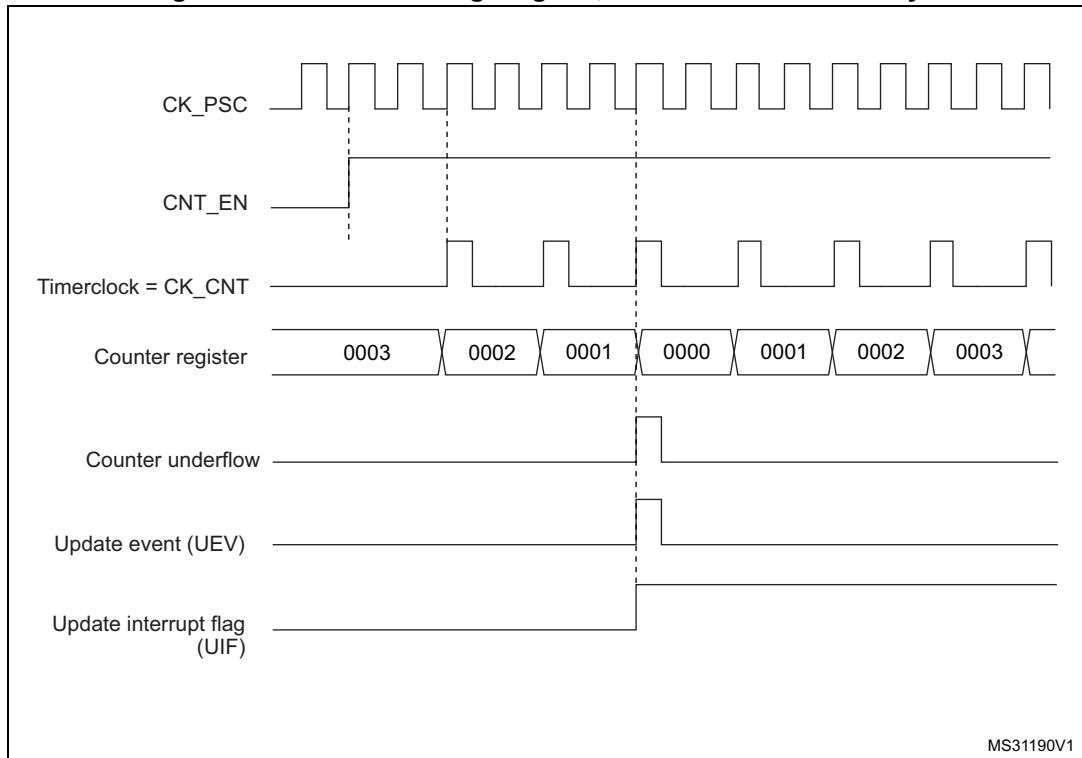
**Figure 175. Counter timing diagram, internal clock divided by 1, TIMx\_ARR=0x6**



1. Here, center-aligned mode 1 is used (for more details refer to [Section 22.4.1: TIMx control register 1 \(TIMx\\_CR1\)\(x = 2 to 4\) on page 662](#)).

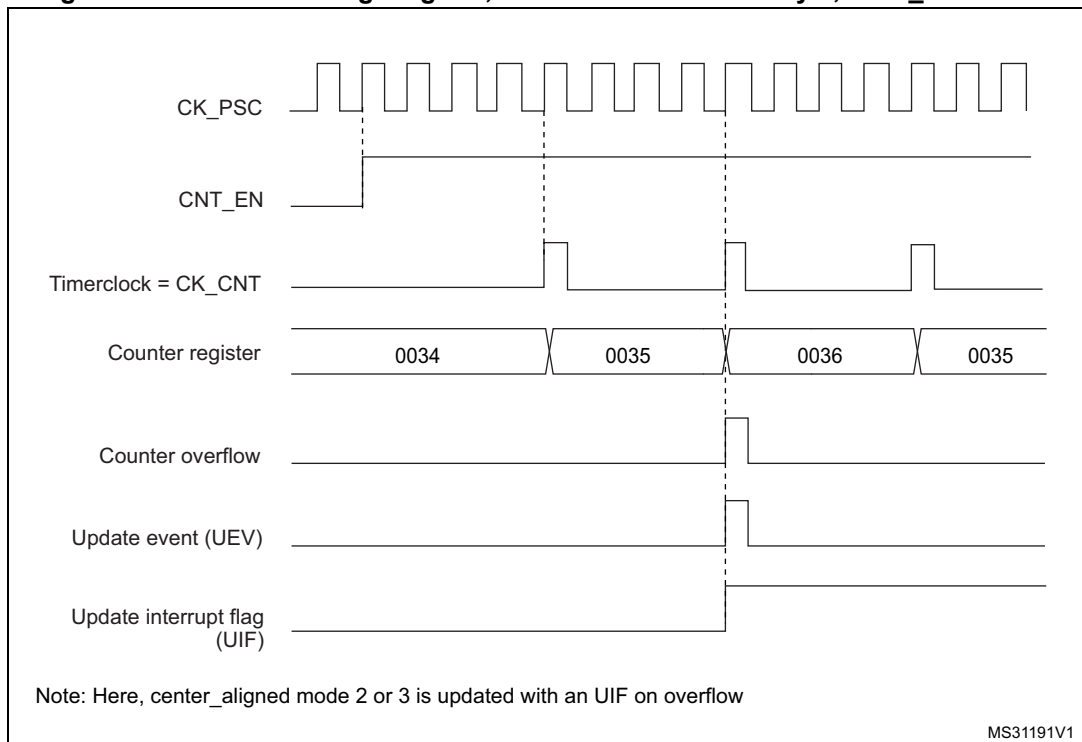


Figure 176. Counter timing diagram, internal clock divided by 2



MS31190V1

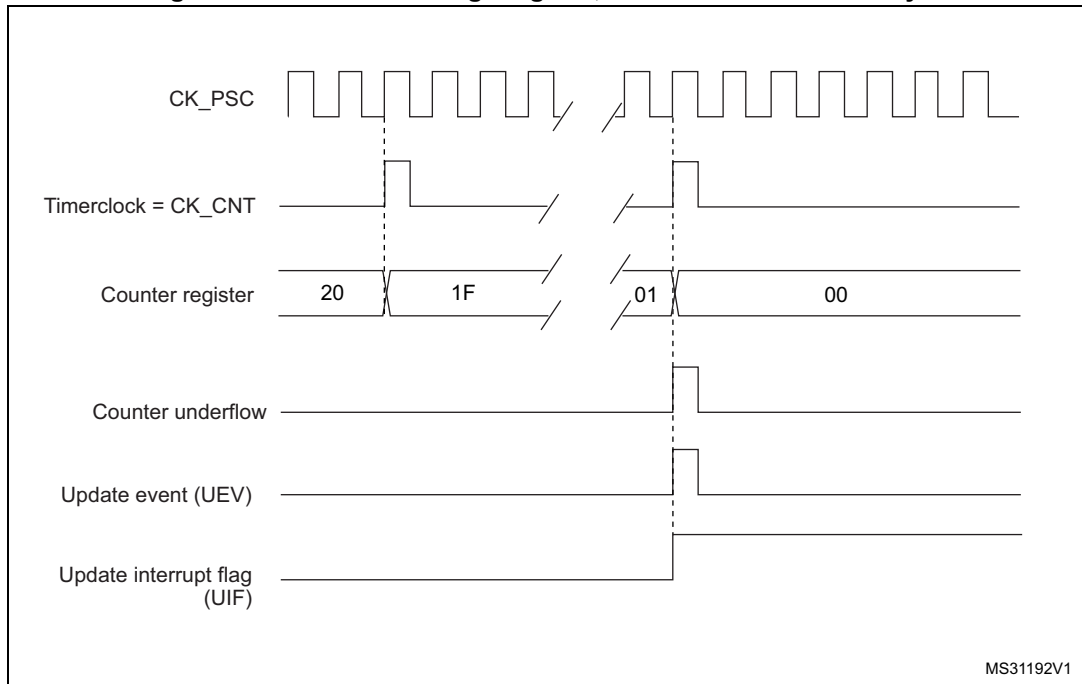
Figure 177. Counter timing diagram, internal clock divided by 4, TIMx\_ARR=0x36



MS31191V1

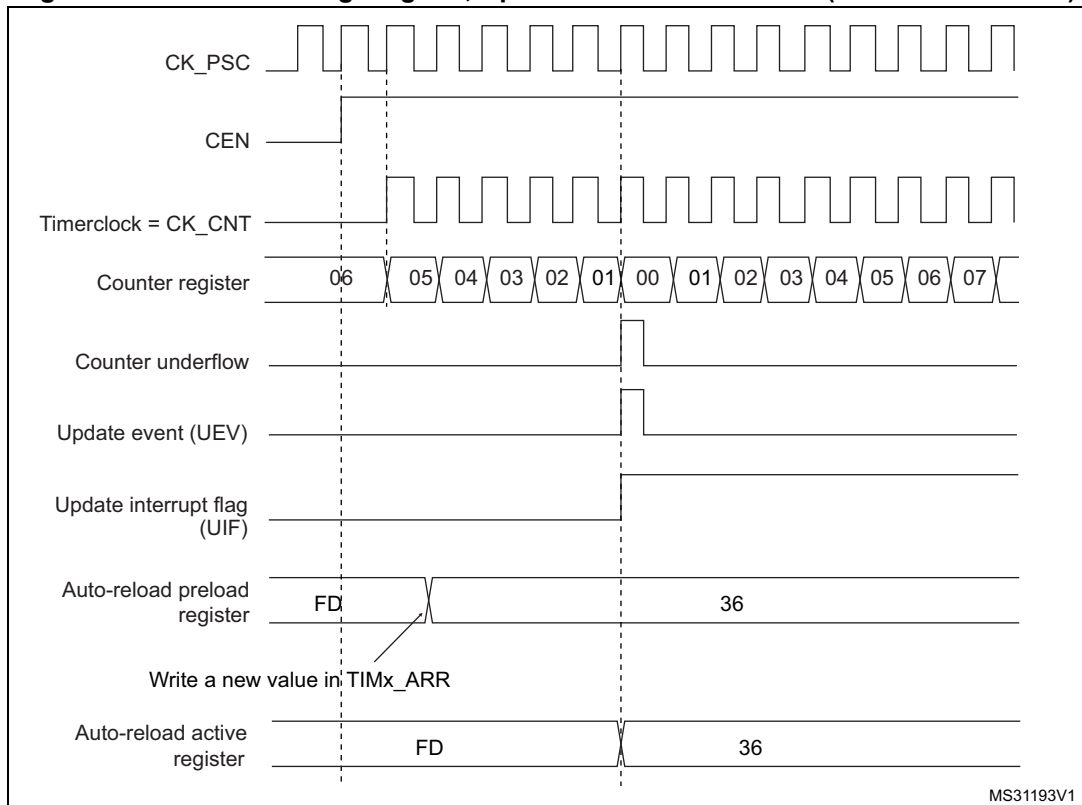
1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 178. Counter timing diagram, internal clock divided by N



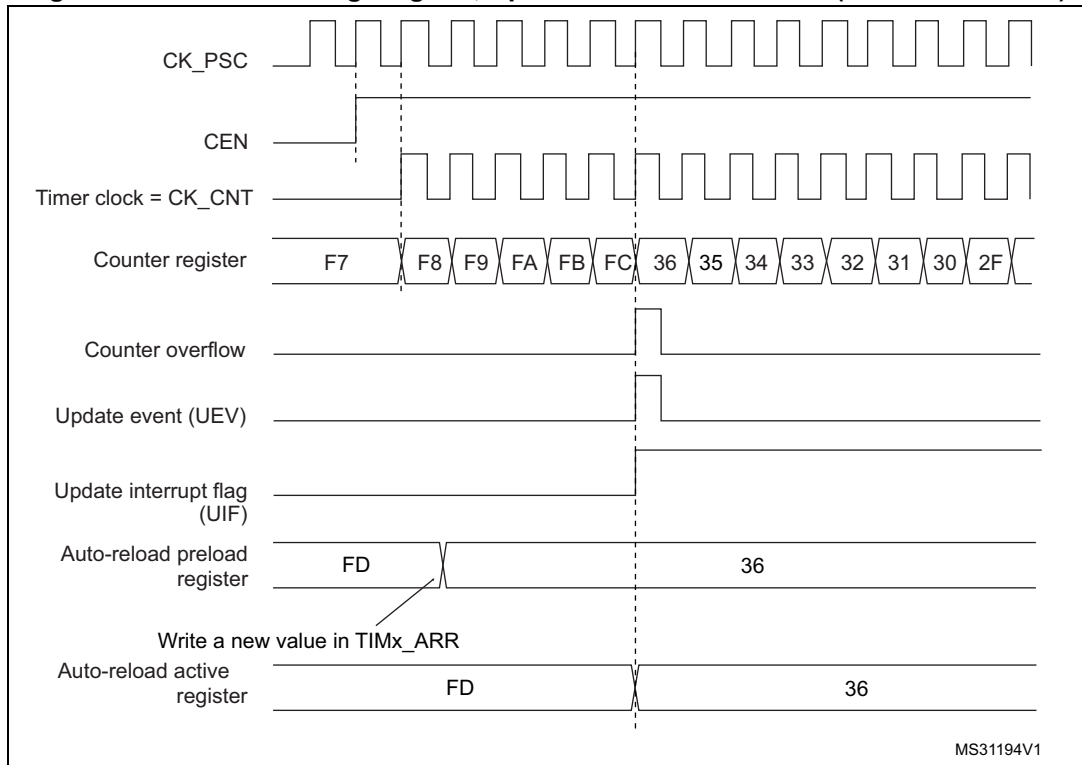
MS31192V1

Figure 179. Counter timing diagram, Update event with ARPE=1 (counter underflow)



MS31193V1

Figure 180. Counter timing diagram, Update event with ARPE=1 (counter overflow)



### 22.3.3 Clock selection

The counter clock can be provided by the following clock sources:

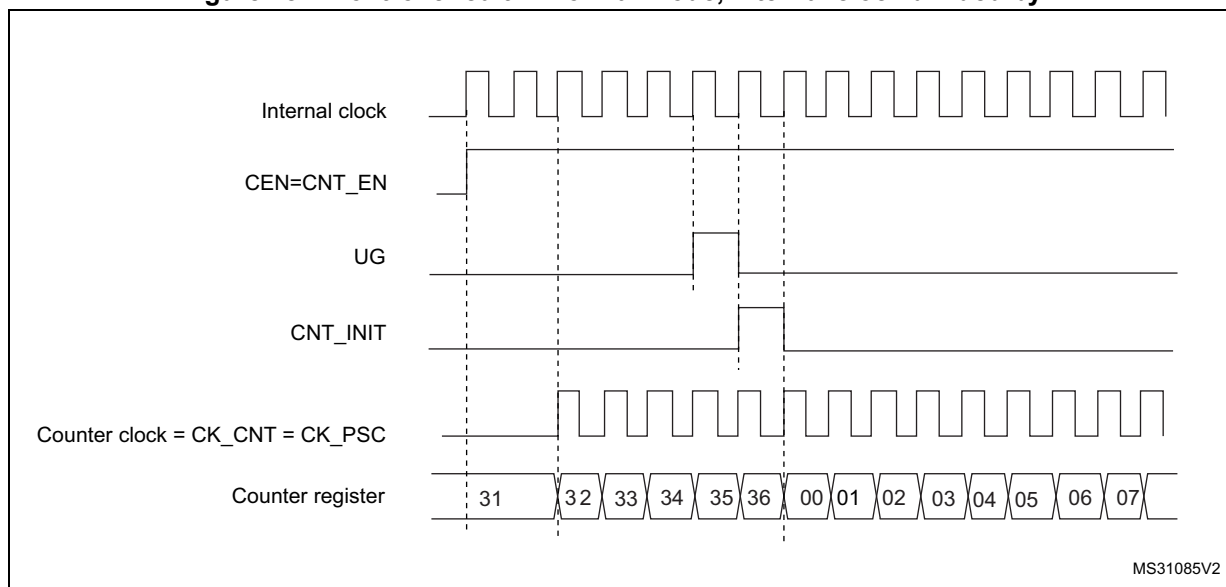
- Internal clock (CK\_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to : [Using one timer as prescaler for another timer on page 656](#) 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 181](#) shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

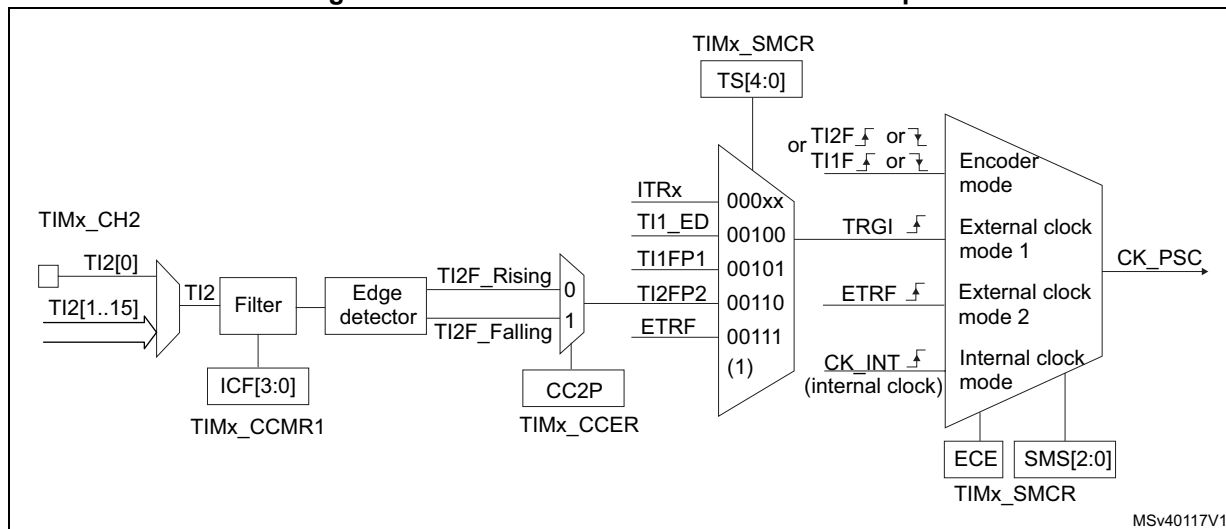
Figure 181. Control circuit in normal mode, internal clock divided by 1



**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 182. TI2 external clock connection example



1. Codes ranging from 01000 to 11111: ITRy.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

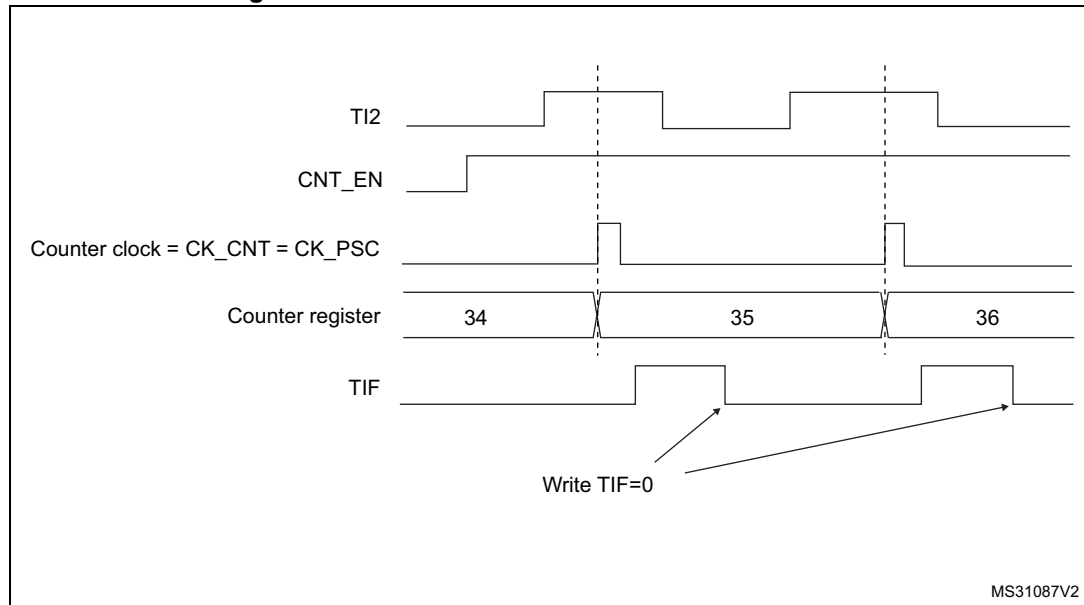
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= '01 in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F=0000).

- Note:* The capture prescaler is not used for triggering, so it does not need to be configured.
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx\_CCER register.
  5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx\_SMCR register.
  6. Select TI2 as the input source by writing TS=00110 in the TIMx\_SMCR register.
  7. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

**Figure 183. Control circuit in external clock mode 1**



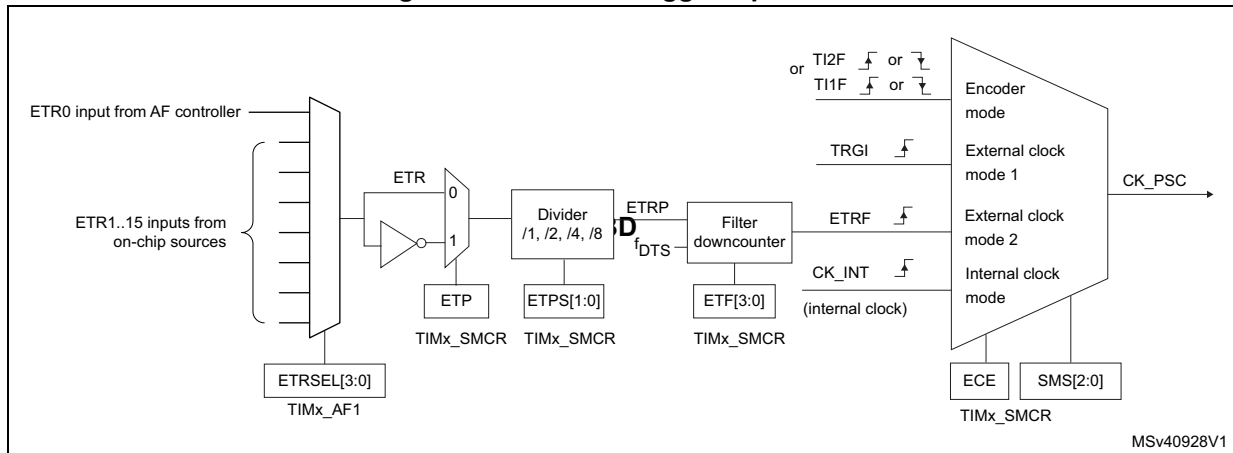
**External clock source mode 2**

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 184](#) gives an overview of the external trigger input block.

Figure 184. External trigger input block



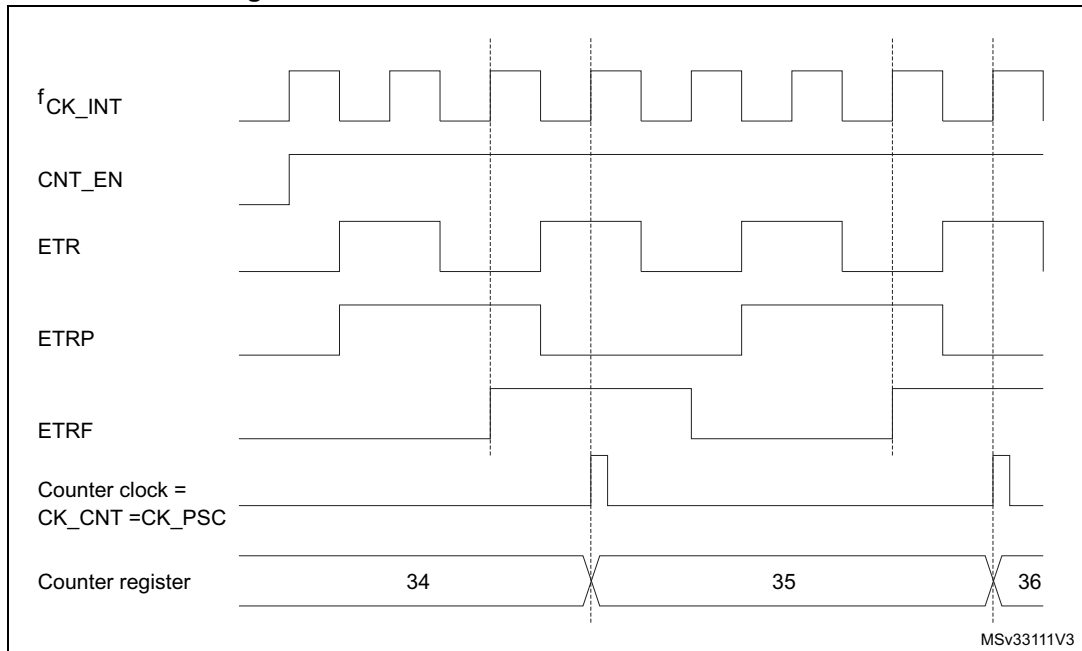
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx\_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx\_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx\_SMCR register
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx\_SMCR register
5. Enable external clock mode 2 by writing ECE=1 in the TIMx\_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most 1/4 of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 185. Control circuit in external clock mode 2



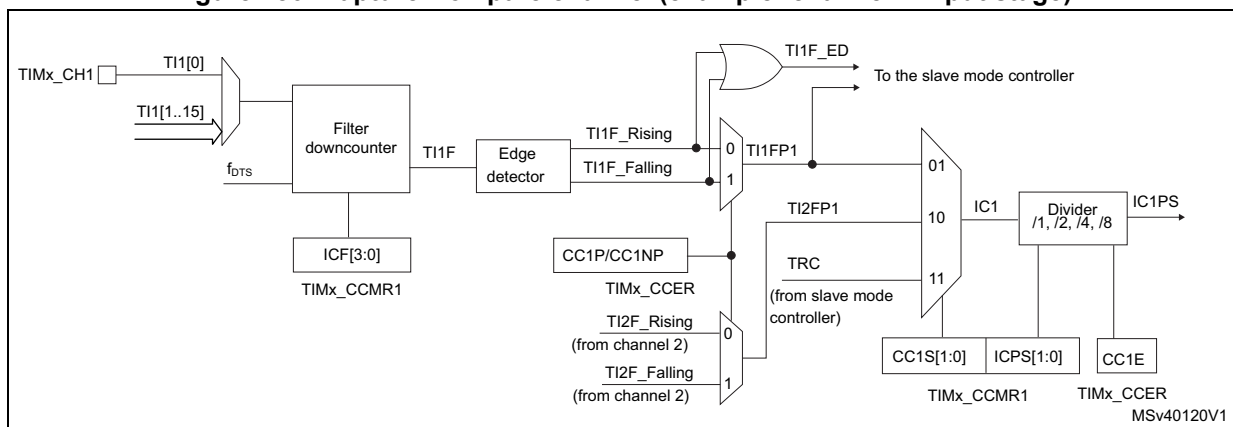
### 22.3.4 Capture/Compare channels

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.

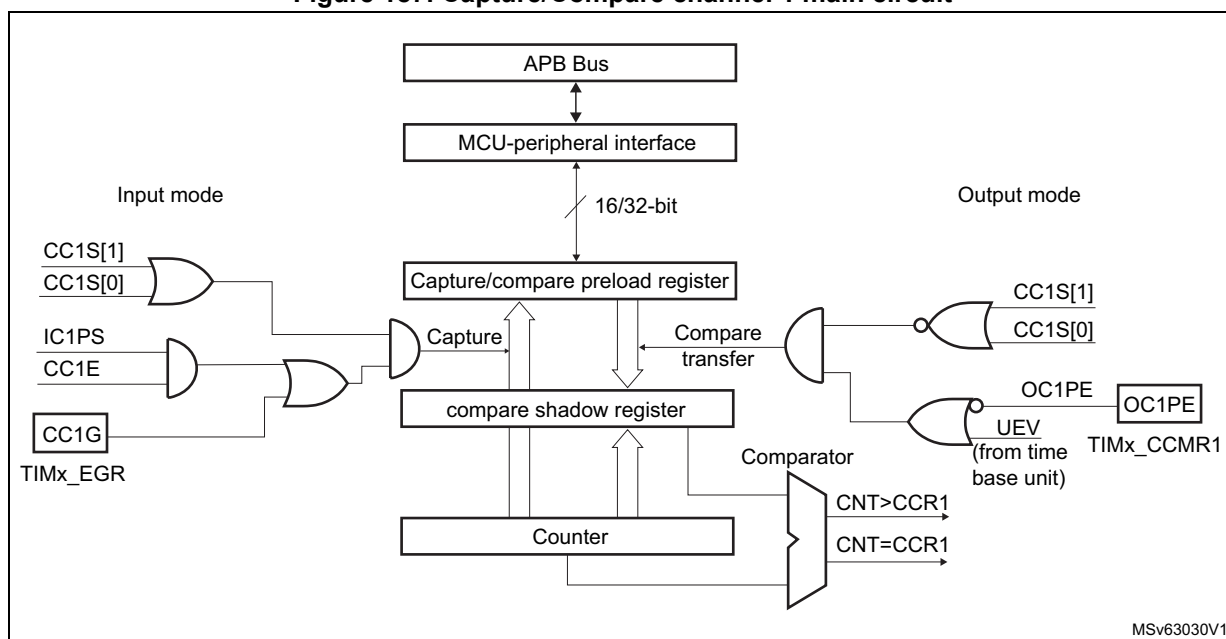
The input stage samples the corresponding  $Tix$  input to generate a filtered signal  $TixF$ . Then, an edge detector with polarity selection generates a signal ( $TixFPx$ ) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register ( $ICxPS$ ).

Figure 186. Capture/Compare channel (example: channel 1 input stage)



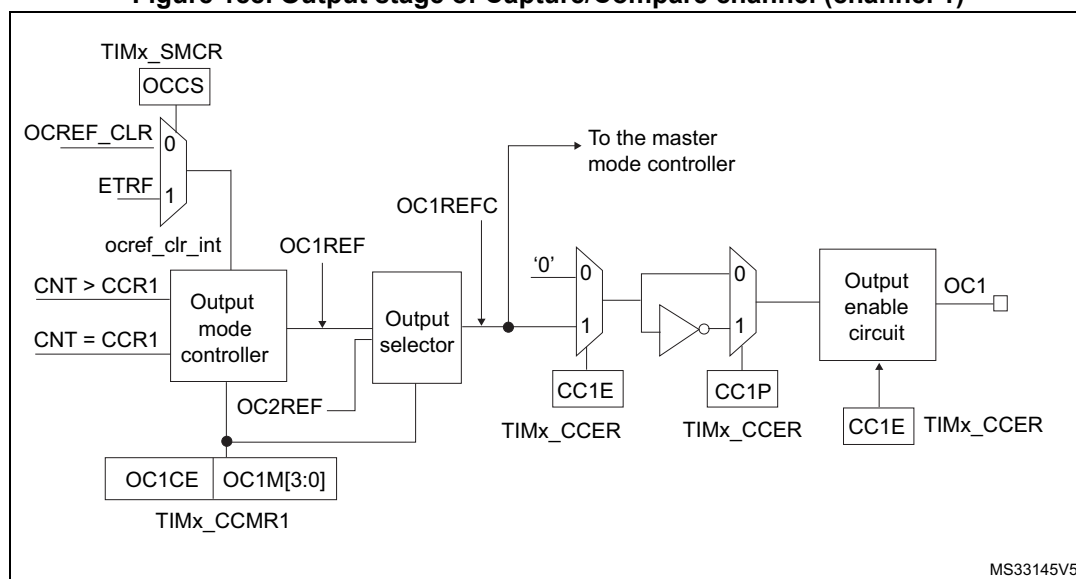
The output stage generates an intermediate waveform which is then used for reference:  $OCxRef$  (active high). The polarity acts at the end of the chain.

Figure 187. Capture/Compare channel 1 main circuit



MSv63030V1

Figure 188. Output stage of Capture/Compare channel (channel 1)



MS33145V5

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.



### 22.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  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 the CC1P and CC1NP bits to 000 in the TIMx\_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx\_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx\_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx\_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx\_DIER register.

When an input capture occurs:

- The TIMx\_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

*Note:* IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx\_EGR register.

### 22.3.6 PWM input mode

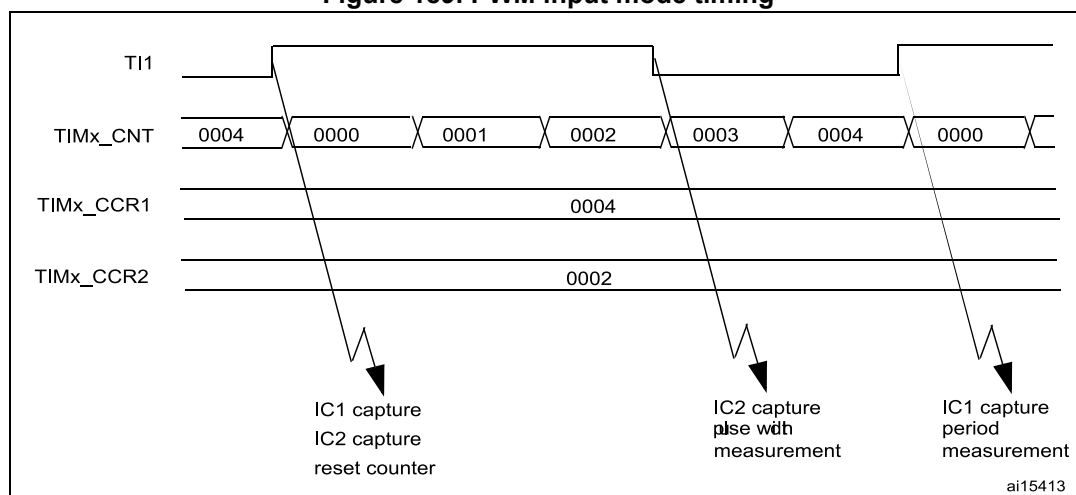
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same T1x input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two T1xFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, 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 189. 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.

### 22.3.7 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.

### 22.3.8 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.

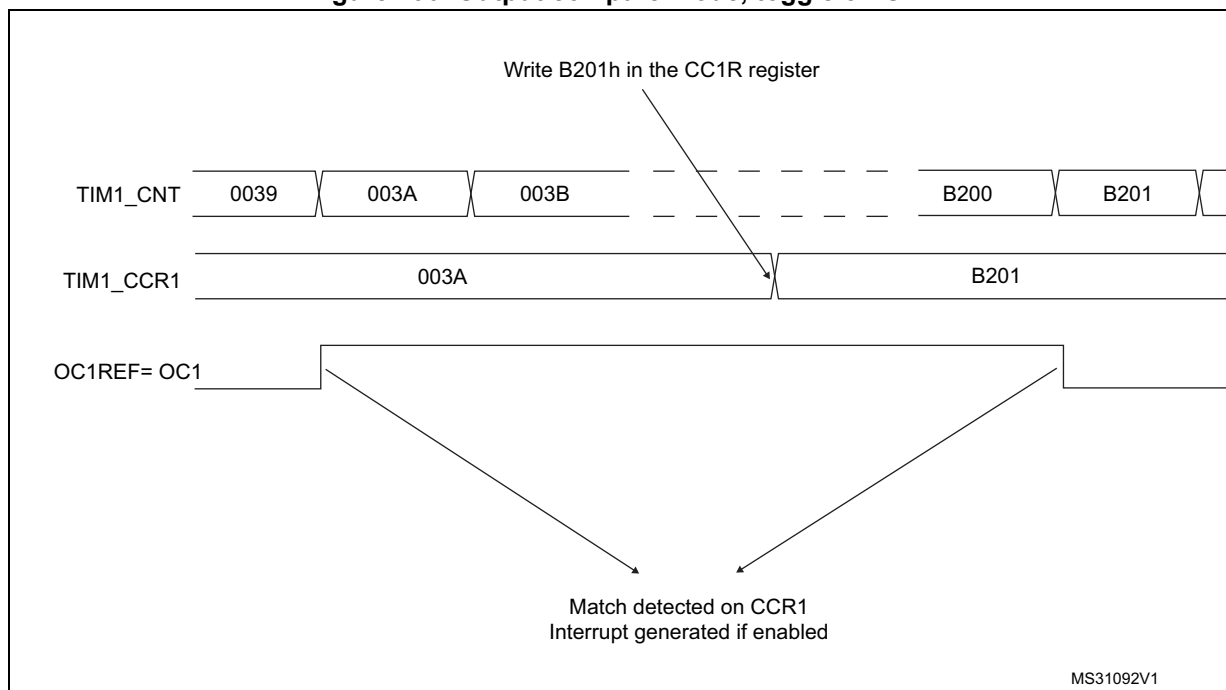
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

#### 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 190](#).

**Figure 190. Output compare mode, toggle on OC1**



### 22.3.9 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.

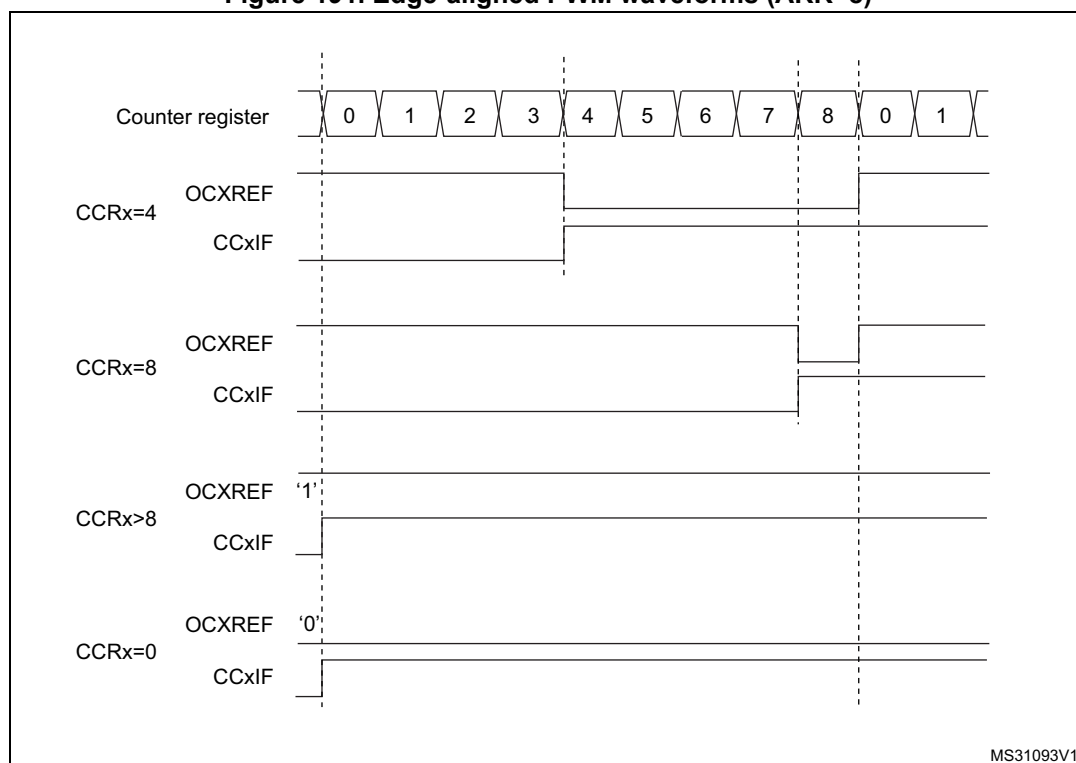
### PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx\_CR1 register is low. Refer to [Upcounting mode on page 621](#).

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 191](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 191. Edge-aligned PWM waveforms (ARR=8)**



### Downcounting configuration

Downcounting is active when DIR bit in TIMx\_CR1 register is high. Refer to [Downcounting mode on page 624](#).

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.

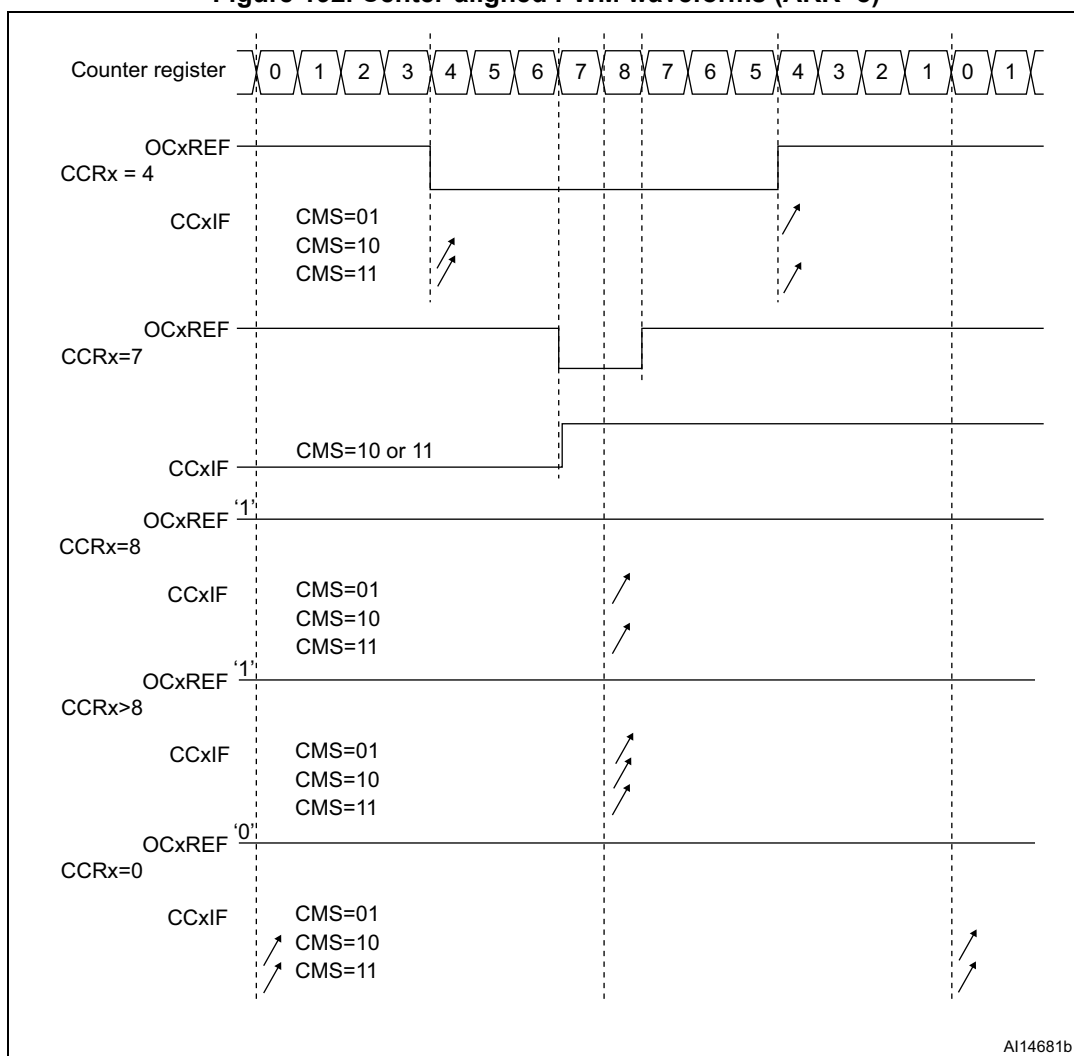
### PWM center-aligned mode

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 627](#).

[Figure 192](#) 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 192. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if 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.

### 22.3.10 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

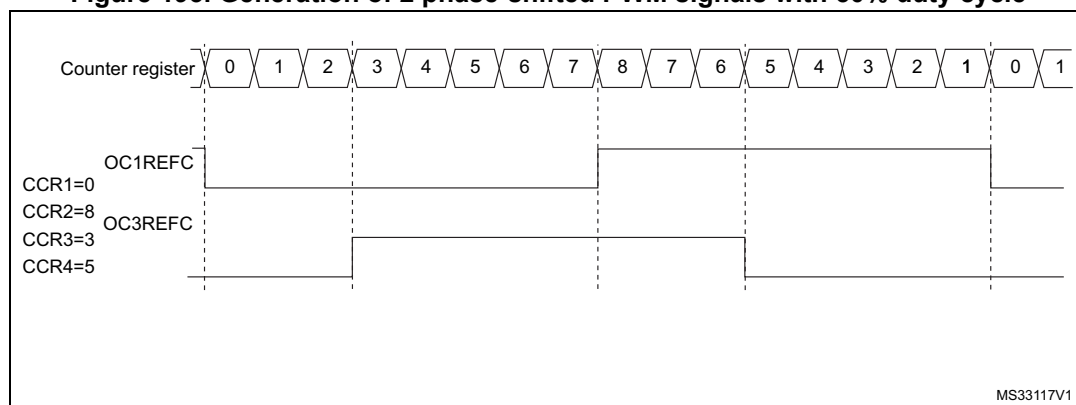
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1110' (Asymmetric PWM mode 1) or '1111' (Asymmetric PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

*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 193 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 193. Generation of 2 phase-shifted PWM signals with 50% duty cycle**



### 22.3.11 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.



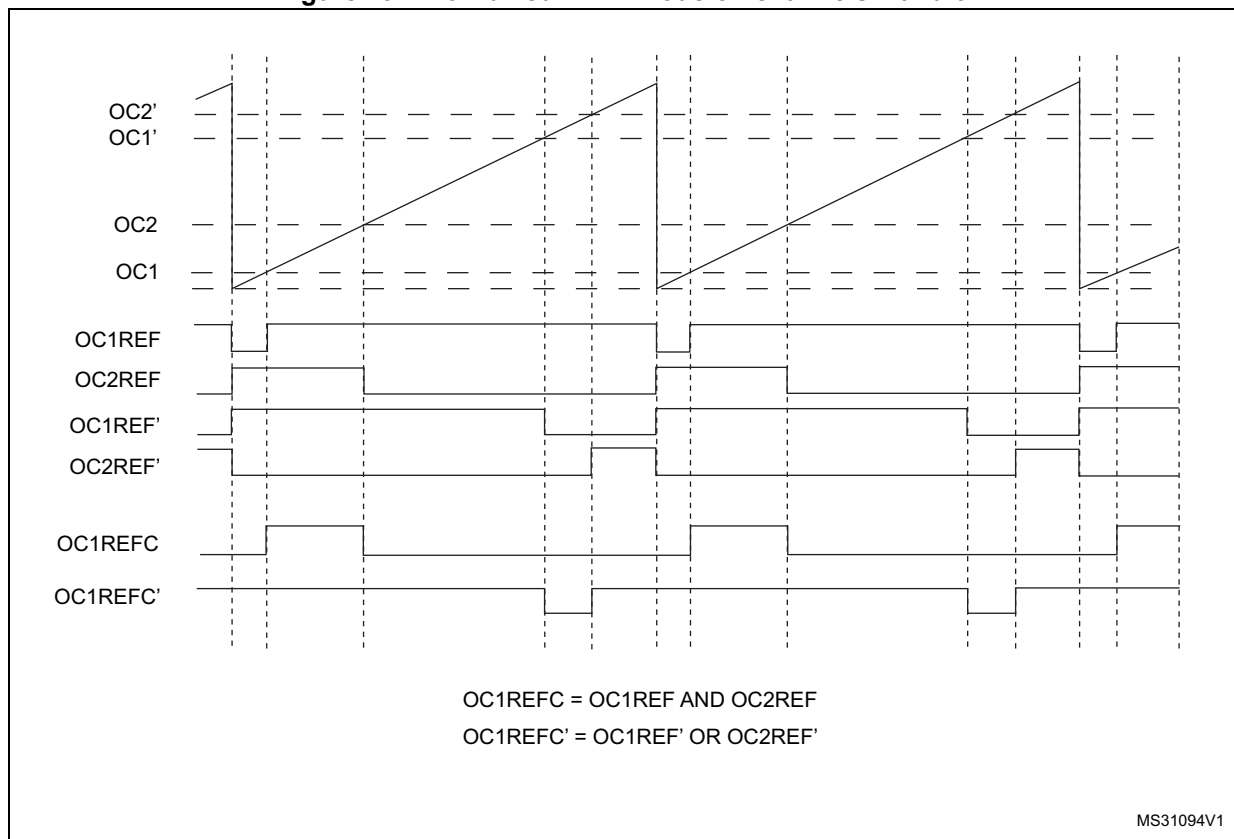
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*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 194 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 194. Combined PWM mode on channels 1 and 3**



### 22.3.12 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.

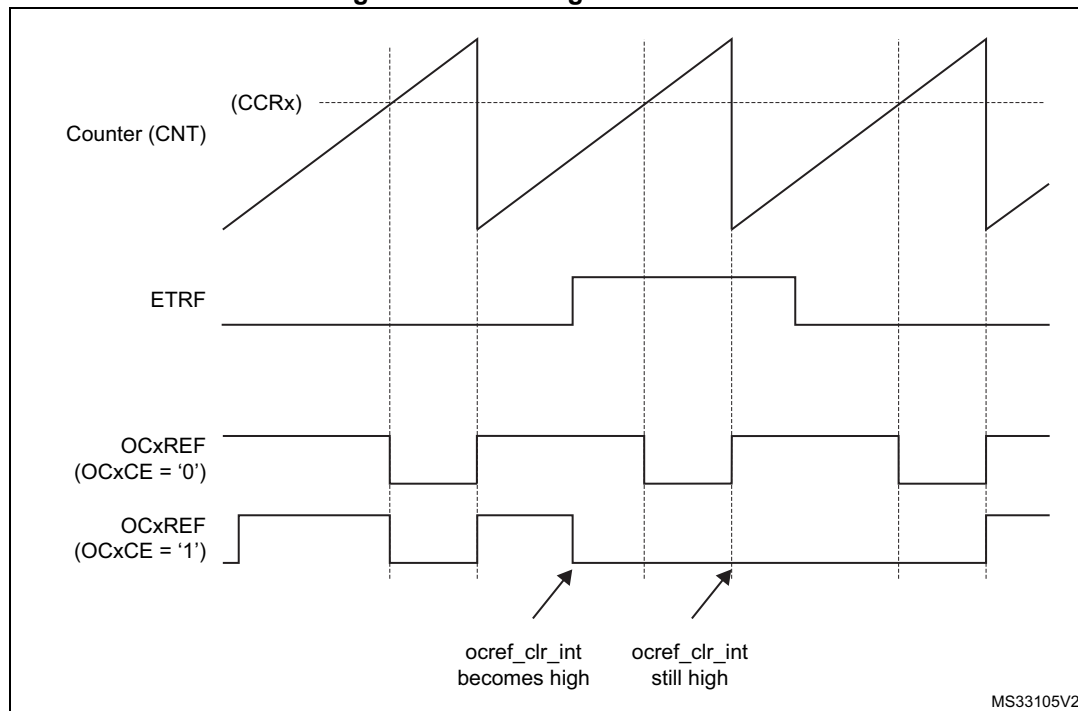
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

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 195 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 195. 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.

### 22.3.13 One-pulse mode

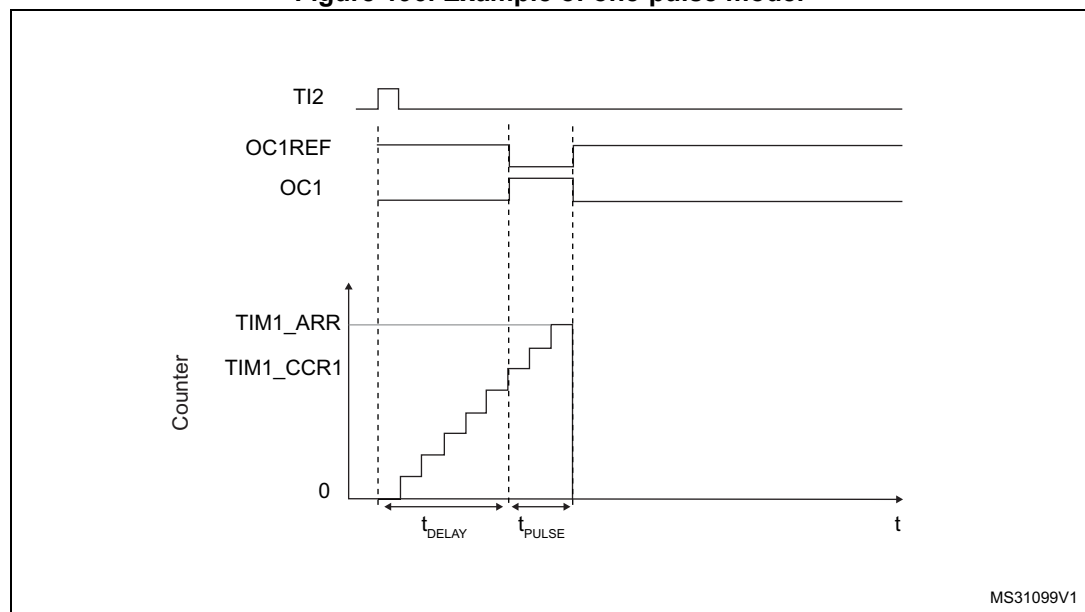
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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:

- $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ ),

**Figure 196. 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  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx\_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case one has to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx\_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx\_CR1 register is set to '0', so the Repetitive Mode is selected.

#### Particular case: OCx fast enable:

In One-pulse mode, the edge detection on Tlx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY min}}$  we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx\_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 22.3.14 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 22.3.13](#):

- 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.

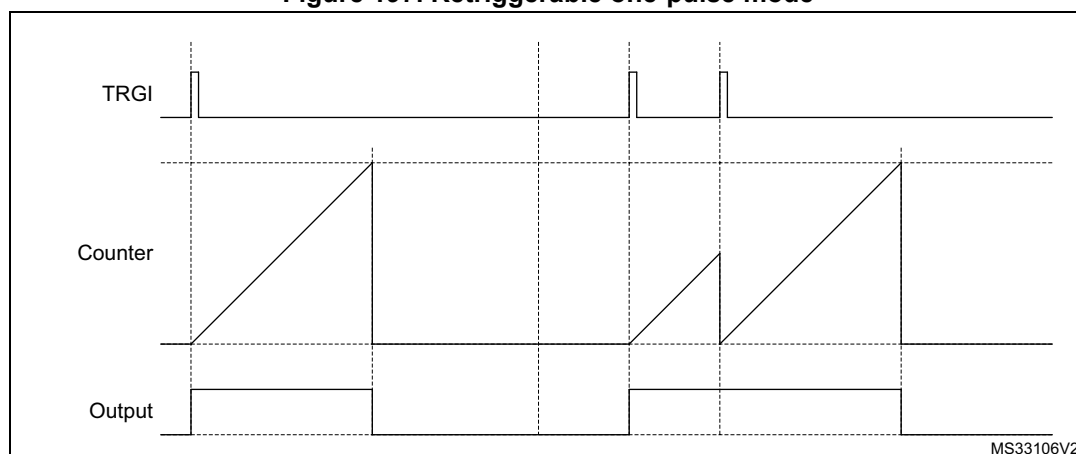
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

*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 197. Retriggerable one-pulse mode



### 22.3.15 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 122](#). 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 122. 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 198 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx\_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx\_CCMR1 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = ‘0’ (TIMx\_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = ‘0’ (TIMx\_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx\_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx\_CR1 register, Counter is enabled)

**Figure 198. Example of counter operation in encoder interface mode**

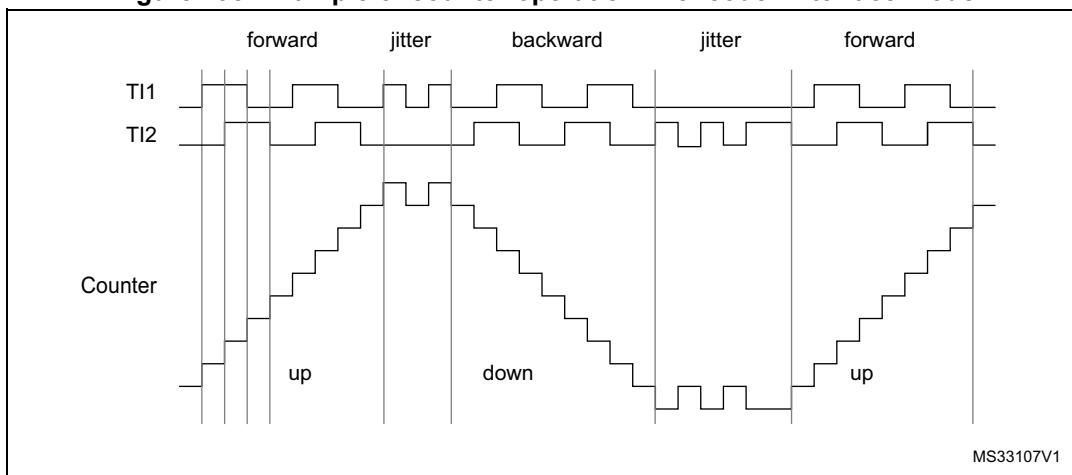
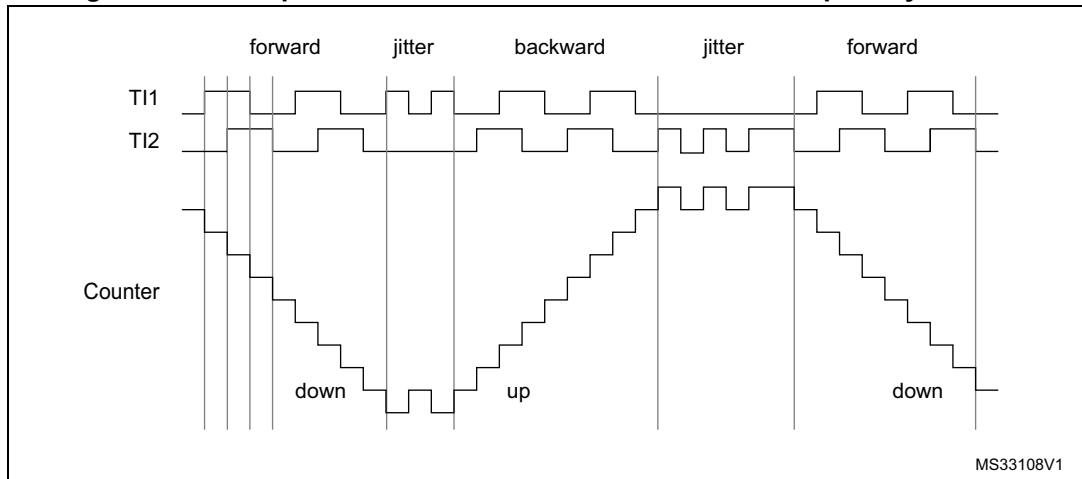


Figure 199 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

**Figure 199. 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.

### 22.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register’s bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

### 22.3.17 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 21.3.25: Interfacing with Hall sensors on page 567](#).

### 22.3.18 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.

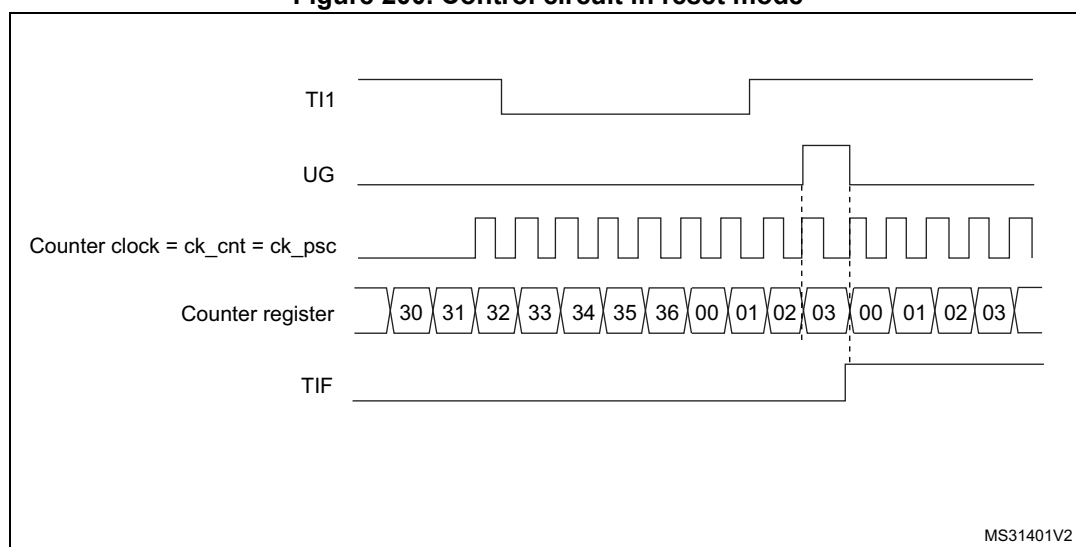
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

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 200. Control circuit in reset mode



#### Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

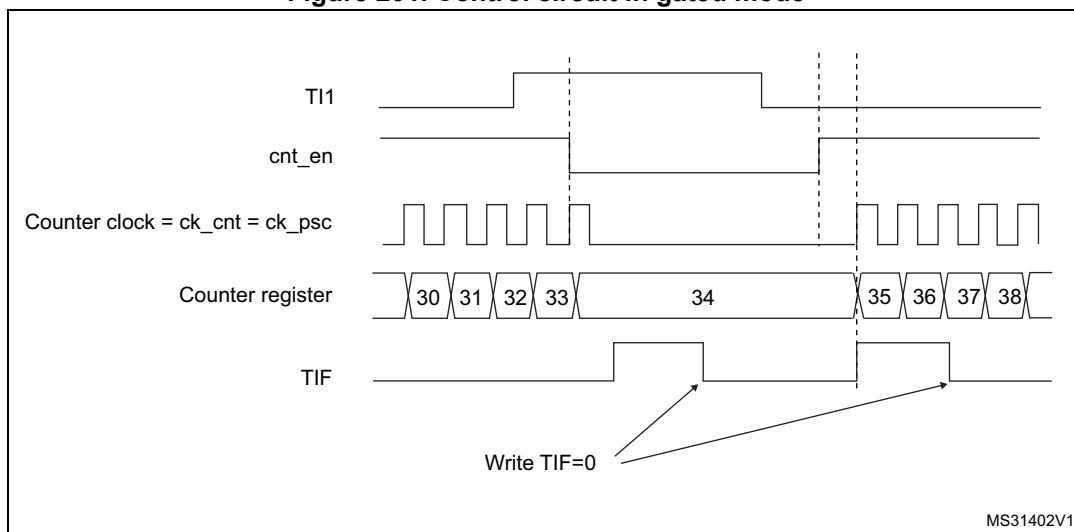


1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 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 201. 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

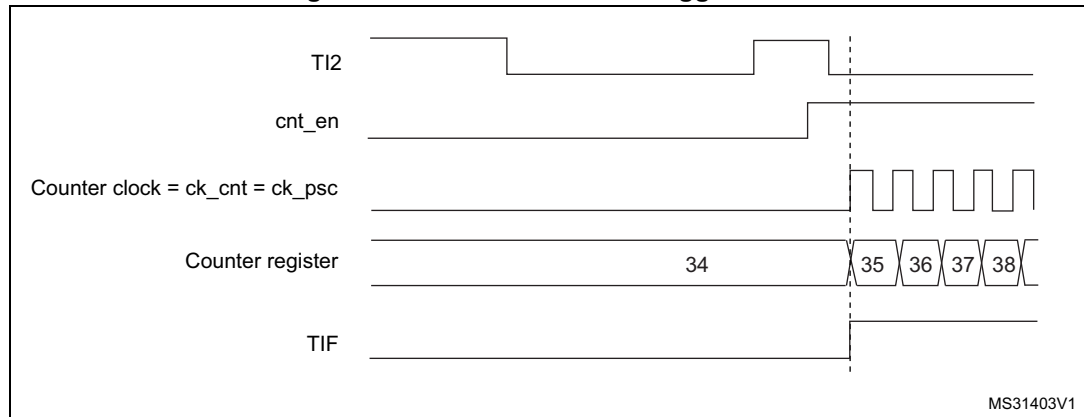
CC2P=1 and CC2NP=0 in TIMx\_CCER register to validate the polarity (and detect low level only).

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 202. 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 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.

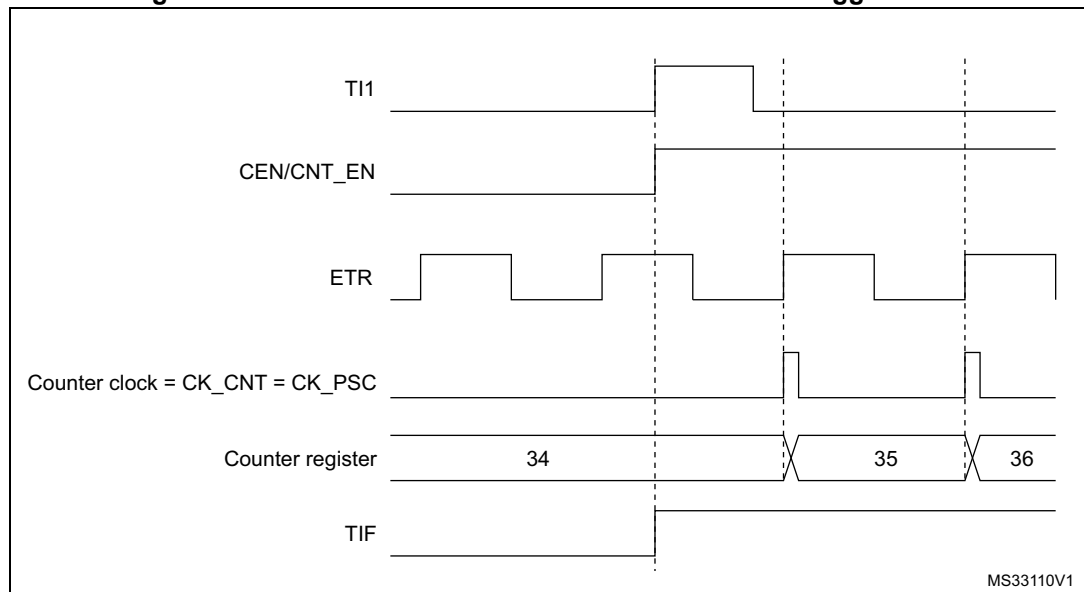
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

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 203. Control circuit in external clock mode 2 + trigger mode**



### 22.3.19 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 204: Master/Slave timer example* and *Figure 205: Master/slave connection example with 1 channel only timers* present an overview of the trigger selection and the master mode selection blocks.

Figure 204. Master/Slave timer example

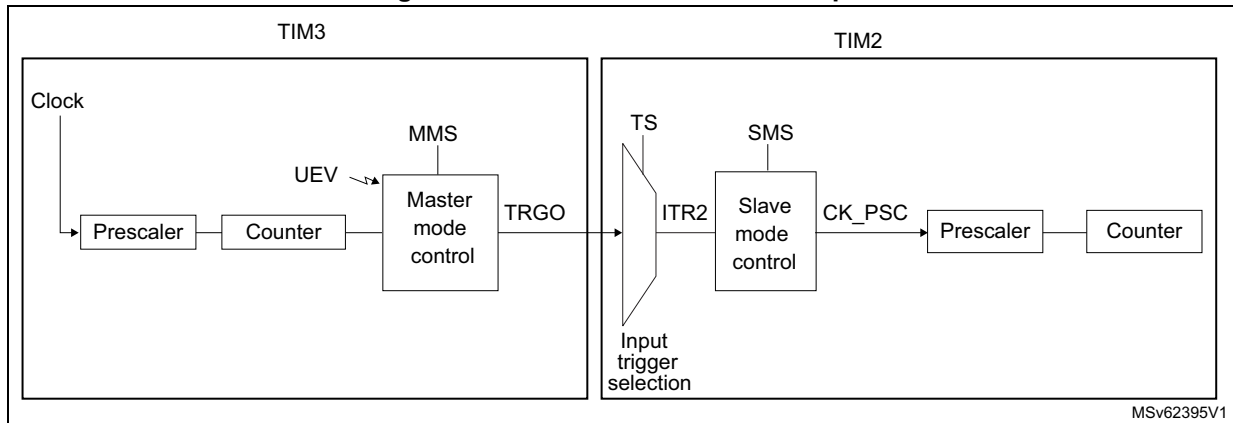
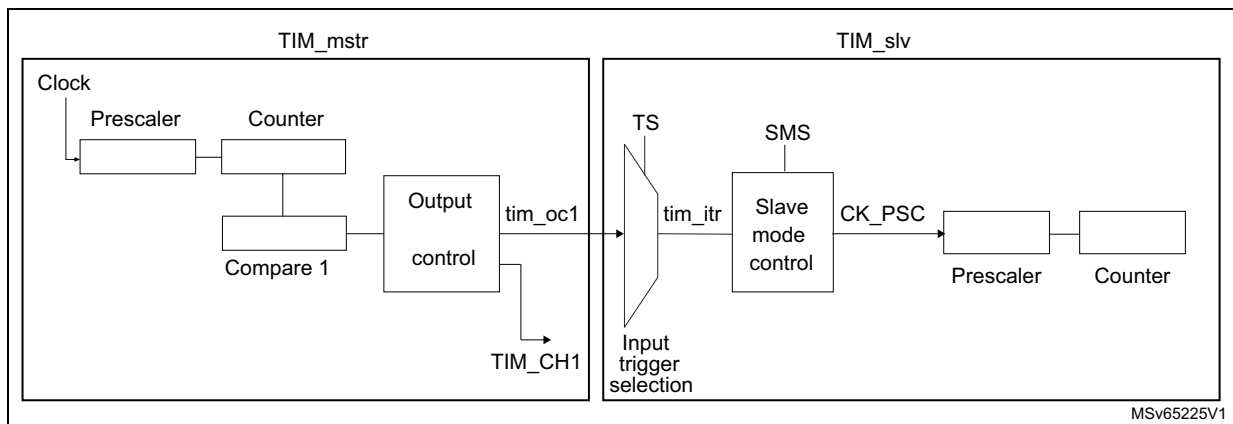


Figure 205. Master/slave connection example with 1 channel only timers



**Note:** The timers with one channel only (see [Figure 205](#)) 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, TIM3 can be configured to act as a prescaler for TIM2. Refer to [Figure 204](#). To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM3\_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM2\_SMCR register (writing TS=00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2\_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx\_CR1 register).

*Note:* If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

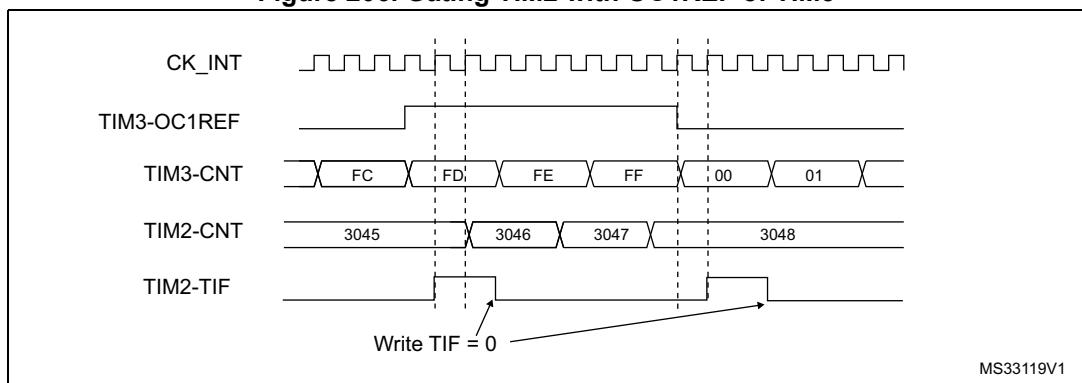
**Using one timer to enable another timer**

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to [Figure 204](#) for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3\_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3\_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2\_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2\_SMCR register).
5. Enable TIM2 by writing '1 in the CEN bit (TIM2\_CR1 register).
6. Start TIM3 by writing '1 in the CEN bit (TIM3\_CR1 register).

*Note:* The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

**Figure 206. Gating TIM2 with OC1REF of TIM3**

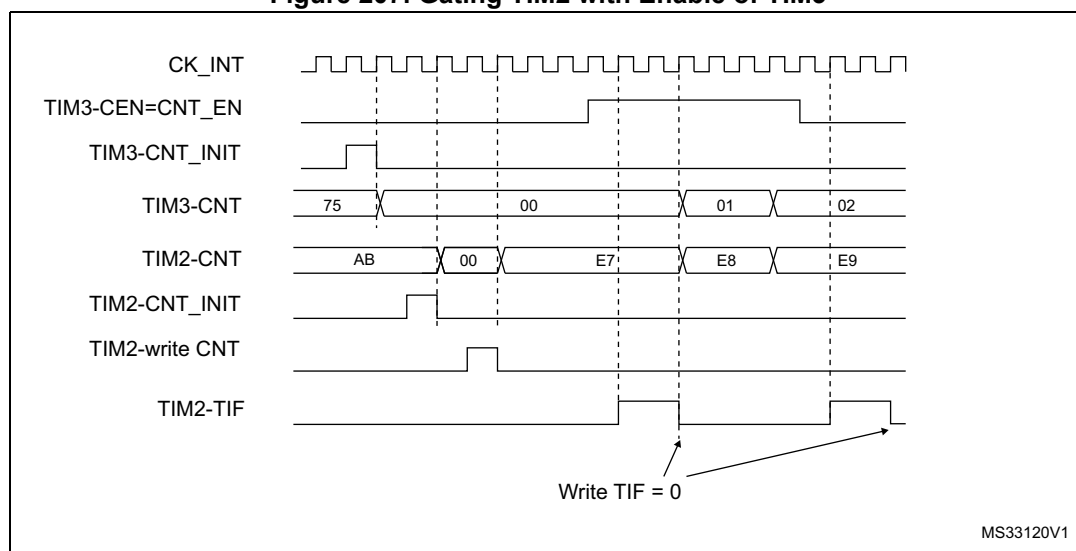


In the example in [Figure 206](#), the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx\_EGR registers.

In the next example (refer to [Figure 207](#)), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing '0 to the CEN bit in the TIM3\_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3\_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3\_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2\_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2\_SMCR register).
5. Reset TIM3 by writing '1 in UG bit (TIM3\_EGR register).
6. Reset TIM2 by writing '1 in UG bit (TIM2\_EGR register).
7. Initialize TIM2 to 0xE7 by writing '0xE7' in the TIM2 counter (TIM2\_CNT).
8. Enable TIM2 by writing '1 in the CEN bit (TIM2\_CR1 register).
9. Start TIM3 by writing '1 in the CEN bit (TIM3\_CR1 register).
10. Stop TIM3 by writing '0 in the CEN bit (TIM3\_CR1 register).

**Figure 207. Gating TIM2 with Enable of TIM3**

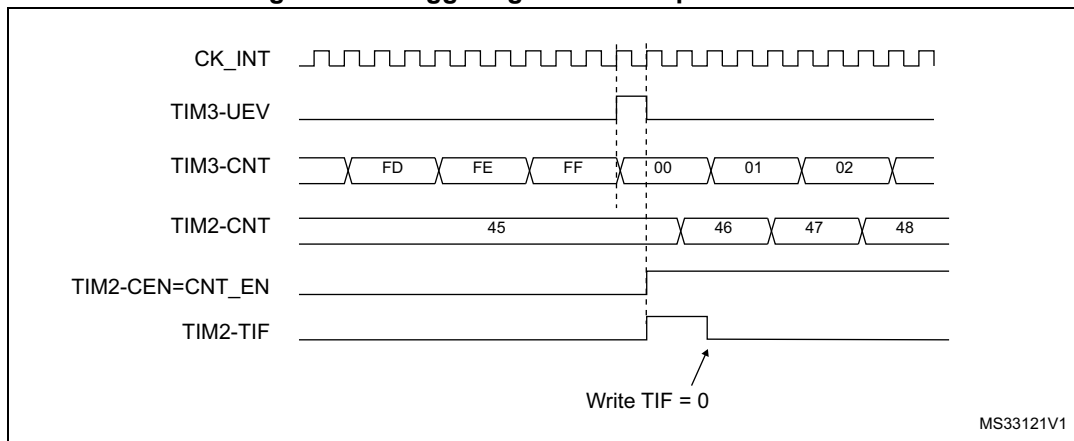


**Using one timer to start another timer**

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to [Figure 204](#) for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write '0 to the CEN bit in the TIM2\_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK\_INT ( $f_{CK\_CNT} = f_{CK\_INT}/3$ ).

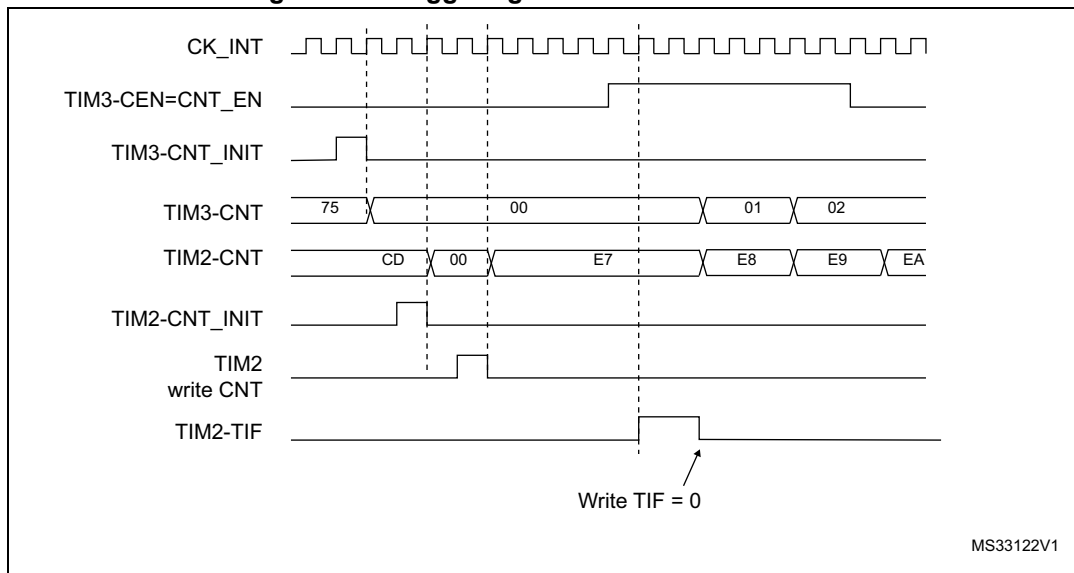
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3\_CR2 register).
2. Configure the TIM3 period (TIM3\_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2\_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2\_SMCR register).
5. Start TIM3 by writing '1 in the CEN bit (TIM3\_CR1 register).

**Figure 208. Triggering TIM2 with update of TIM3**



As in the previous example, both counters can be initialized before starting counting. [Figure 209](#) shows the behavior with the same configuration as in [Figure 208](#) but in trigger mode instead of gated mode (SMS=110 in the TIM2\_SMCR register).

**Figure 209. Triggering TIM2 with Enable of TIM3**



**Starting 2 timers synchronously in response to an external trigger**

In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to [Figure 204](#) for connections. To ensure the counters are

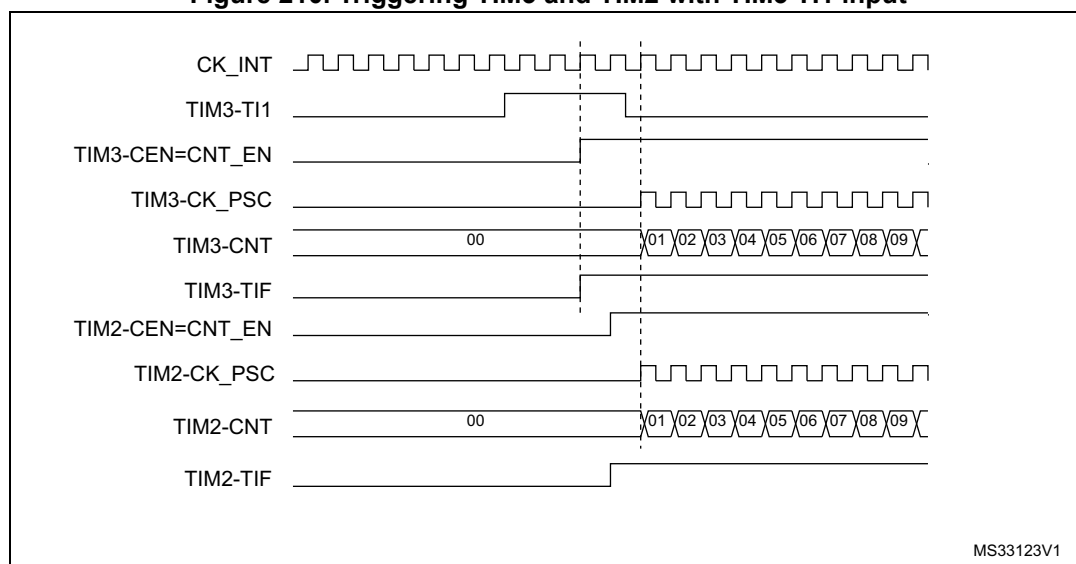
aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3\_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=00100 in the TIM3\_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3\_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3\_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=00000 in the TIM2\_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2\_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

*Note:* In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx\_CNT). One can see that the master/slave mode insert a delay between CNT\_EN and CK\_PSC on TIM3.

**Figure 210. Triggering TIM3 and TIM2 with TIM3 TI1 input**



*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.

### 22.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.



The DMA controller destination is unique and must point to the virtual register TIMx\_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx\_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx\_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx\_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register:

Example:

00000: TIMx\_CR1

00001: TIMx\_CR2

00010: TIMx\_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
  - DMA channel peripheral address is the DMAR register address
  - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
  - Number of data to transfer = 3 (See note below).
  - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:  
DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 22.3.21 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 in DBG module. For more details, refer to [Section 40.9.2: Debug support for timers, watchdog and I<sup>2</sup>C](#).

## 22.4 TIM2/TIM3/TIM4 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).

### 22.4.1 TIMx control register 1 (TIMx\_CR1)(x = 2 to 4)

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.

### 22.4.2 TIMx control register 2 (TIMx\_CR2)(x = 2 to 4)

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.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: 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 21.3.25: Interfacing with Hall sensors on page 567](#)

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.

**22.4.3 TIMx slave mode control register (TIMx\_SMCR)(x = 2 to 4)**

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 123: TIMx internal trigger connection on page 668](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source

0: OCREF\_CLR\_INT is connected to COMP1, COMP2 or COMP3 output depending on TIMx\_OR1.OCREF\_CLR[1:0]

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 123. TIMx internal trigger connection**

Slave TIM	ITR0	ITR1	ITR2	ITR3
TIM2	TIM1	TIM15	TIM3	TIM14_OC1
TIM3	TIM1	TIM2	TIM15	TIM14_OC1
TIM4	TIM1	TIM2	TIM15	TIM14_OC1

### 22.4.4 TIMx DMA/Interrupt enable register (TIMx\_DIER)(x = 2 to 4)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled.

1: Trigger DMA request enabled.

Bit 13 Reserved, 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.

### 22.4.5 TIMx status register (TIMx\_SR)(x = 2 to 4)

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

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.

### 22.4.6 TIMx event generation register (TIMx\_EGR)(x = 2 to 4)

Address offset: 0x14

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.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

- Bit 6 **TG**: Trigger generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled.
- Bit 5 Reserved, must be kept at reset value.
- Bit 4 **CC4G**: Capture/compare 4 generation  
Refer to CC1G description
- Bit 3 **CC3G**: Capture/compare 3 generation  
Refer to CC1G description

- Bit 2 **CC2G**: Capture/compare 2 generation  
Refer to CC1G description
- Bit 1 **CC1G**: Capture/compare 1 generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: A capture/compare event is generated on channel 1:  
**If channel CC1 is configured as output:**  
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.  
**If channel CC1 is configured as input:**  
The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.
- Bit 0 **UG**: Update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx\_ARR) if DIR=1 (downcounting).

### 22.4.7 TIMx capture/compare mode register 1 (TIMx\_CCMR1)(x = 2 to 4)

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).*

**22.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx\_CCMR1) (x = 2 to 4)**

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.	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).*

### 22.4.9 TIMx capture/compare mode register 2 (TIMx\_CCMR2)(x = 2 to 4)

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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler



Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection  
 This bit-field defines the direction of the channel (input/output) as well as the used input.  
 00: CC4 channel is configured as output  
 01: CC4 channel is configured as input, IC4 is mapped on TI4  
 10: CC4 channel is configured as input, IC4 is mapped on TI3  
 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)  
*Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx\_CCER).*

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection  
 This bit-field defines the direction of the channel (input/output) as well as the used input.  
 00: CC3 channel is configured as output  
 01: CC3 channel is configured as input, IC3 is mapped on TI3  
 10: CC3 channel is configured as input, IC3 is mapped on TI4  
 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)  
*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

### 22.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx\_CCMR2) (x = 2 to 4)

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 enable
- Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode  
 Refer to OC1M description (bits 6:4 in TIMx\_CCMR1 register)
- Bit 3 **OC3PE**: Output compare 3 preload enable
- Bit 2 **OC3FE**: Output compare 3 fast enable
- Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection  
 This bit-field defines the direction of the channel (input/output) as well as the used input.  
 00: CC3 channel is configured as output  
 01: CC3 channel is configured as input, IC3 is mapped on TI3  
 10: CC3 channel is configured as input, IC3 is mapped on TI4  
 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)  
*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx\_CCER).*

### 22.4.11 TIMx capture/compare enable register (TIMx\_CCER)(x = 2 to 4)

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
r/w		r/w	r/w	r/w		r/w	r/w	r/w		r/w	r/w	r/w		r/w	r/w

- Bit 15 **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 T11FP1/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 T11FP1 and TI2FP1 for trigger or capture operations.
  - CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to T1xFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is not inverted (trigger operation in gated mode or encoder mode).
  - CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to T1xFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is inverted (trigger operation in gated mode or encoder mode).
  - CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both T1xFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 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 124. 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.

**22.4.12 TIMx counter (TIMx\_CNT)(x = 2 to 4)**

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
CNT[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
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CNT[31:16]**: Most significant part counter value (TIM2)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

### 22.4.13 TIMx counter [alternate] (TIMx\_CNT)(x = 2 to 4)

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	CNT[30:16]														
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
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 **CNT[30:16]**: Most significant part counter value (TIM2)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

### 22.4.14 TIMx prescaler (TIMx\_PSC)(x = 2 to 4)

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”).

### 22.4.15 TIMx auto-reload register (TIMx\_ARR)(x = 2 to 4)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

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

Bits 31:16 **ARR[31:16]**: High auto-reload value (TIM2)

Bits 15:0 **ARR[15:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 22.3.1: Time-base unit on page 619](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 22.4.16 TIMx capture/compare register 1 (TIMx\_CCR1)(x = 2 to 4)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (TIM2)

Bits 15:0 **CCR1[15:0]**: Low Capture/Compare 1 value

**If channel CC1 is configured as output:**

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**

CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx\_CCR1 register is read-only and cannot be programmed.

### 22.4.17 TIMx capture/compare register 2 (TIMx\_CCR2)(x = 2 to 4)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (TIM2)

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

**If channel CC2 is configured as output:**

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and 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.

### 22.4.18 TIMx capture/compare register 3 (TIMx\_CCR3)(x = 2 to 4)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (TIM2)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

**If channel CC3 is configured as output:**

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC3 output.

**If channel CC3 is configured as input:**

CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx\_CCR3 register is read-only and cannot be programmed.

**22.4.19 TIMx capture/compare register 4 (TIMx\_CCR4)(x = 2 to 4)**

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (TIM2)

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

- 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.
- 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.

### 22.4.20 TIMx DMA control register (TIMx\_DCR)(x = 2 to 4)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx\_DMAR address).

- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx\_DMAR address). DBA is defined as an offset starting from the address of the TIMx\_CR1 register.

Example:

- 00000: TIMx\_CR1
- 00001: TIMx\_CR2
- 00010: TIMx\_SMCR
- ...

**Example:** Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx\_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx\_CR1 address.

### 22.4.21 TIMx DMA address for full transfer (TIMx\_DMAR)(x = 2 to 4)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address  $(\text{TIMx\_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 22.4.22 TIM2 option register 1 (TIM2\_OR1)

Address offset: 0x50

Reset value: 0x0000 0000



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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.	OCREF_CLR [1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OCREF\_CLR[1:0]**: Ocref\_clr source selection

This bit selects the ocref\_clr input source.

00: COMP1 output is connected to the OCREF\_CLR input

01: COMP2 output is connected to the OCREF\_CLR input

10: COMP3<sup>(1)</sup> output is connected to the OCREF\_CLR input

11: Reserved

1. COMP3 is available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 22.4.23 TIM3 option register 1 (TIM3\_OR1)

Address offset: 0x50

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.	Res.	Res.	OCREF_CLR [1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OCREF\_CLR[1:0]**: Ocref\_clr source selection

This bit selects the ocref\_clr input source.

Bits 1:0 OCREF\_CLR[1:0]

00: COMP1 output is connected to the OCREF\_CLR input

01: COMP2 output is connected to the OCREF\_CLR input

10: COMP3<sup>(1)</sup> output is connected to the OCREF\_CLR input

11: Reserved

1. COMP3 is available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 22.4.24 TIM4 option register 1 (TIM4\_OR1)

Address offset: 0x50

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.	Res.	Res.	OCREF_CLR [1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **OCREF\_CLR[1:0]**: Ocref\_clr source selection

This bit selects the ocref\_clr input source.

Bits 1:0 OCREF\_CLR[1:0]

00: COMP1 output is connected to the OCREF\_CLR input

01: COMP2 output is connected to the OCREF\_CLR input

10: COMP3<sup>(1)</sup> output is connected to the OCREF\_CLR input

11: Reserved

1. COMP3 is available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 22.4.25 TIM2 alternate function option register 1 (TIM2\_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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

0001: COMP1

0010: COMP2

0011: LSE

0100: MCO<sup>(1)</sup>

0101: MCO2<sup>(1)</sup>

0110: COMP3<sup>(1)</sup>

Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

1. Available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 22.4.26 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

0001: COMP1 output

0010: COMP2 output

0110: COMP3 output<sup>(1)</sup>

Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

1. Available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 22.4.27 TIM4 alternate function option register 1 (TIM4\_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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

0001: COMP1 output

0010: COMP2 output

0110: COMP3 output<sup>(1)</sup>

Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

1. Available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 22.4.28 TIM2 timer input selection register (TIM2\_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.	T13SEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	T12SEL[3:0]				Res.	Res.	Res.	Res.	T11SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 19:16 **T13SEL[3:0]**: T13[0] to T13[15] input selection

These bits select the T13[0] to T13[15] input source.

0000: TIM2\_CH3 input

0001: COMP3 output

Others: Reserved

*Note: Available on STM32G0B1xx and STM32G0C1xx salestypes only, otherwise reserved.*

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

Bits 11:8 **T12SEL[3:0]**: T12[0] to T12[15] input selection

These bits select the T12[0] to T12[15] input source.

0000: TIM2\_CH2 input

0001: COMP2 output

Others: Reserved

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

Bits 3:0 **T11SEL[3:0]**: T11[0] to T11[15] input selection

These bits select the T11[0] to T11[15] input source.

0000: TIM2\_CH1 input

0001: COMP1 output

Others: Reserved

### 22.4.29 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.	T13SEL[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	T12SEL[3:0]				Res.	Res.	Res.	Res.	T11SEL[3:0]			
				rw	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  
 0001: COMP3 output  
 Others: Reserved

*Note: Available on STM32G0B1xx and STM32G0C1xx salestypes only, otherwise 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  
 0001: COMP2 output  
 Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection  
 These bits select the TI1[0] to TI1[15] input source.  
 0000: TIM3\_CH1 input  
 0001: COMP1 output  
 Others: Reserved

### 22.4.30 TIM4 timer input selection register (TIM4\_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.	TI3SEL[3:0]			
												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 31:20 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: TIM4\_CH3 input  
 0001: COMP3 output  
 Others: Reserved

*Note: Available on STM32G0B1xx and STM32G0C1xx salestypes only, otherwise reserved.*

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

Bits 11:8 **TI2SEL[3:0]**: T12[0] to T12[15] input selection

These bits select the T12[0] to T12[15] input source.

0000: TIM4\_CH2 input

0001: COMP2 output

Others: Reserved

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

Bits 3:0 **TI1SEL[3:0]**: T11[0] to T11[15] input selection

These bits select the T11[0] to T11[15] input source.

0000: TIM4\_CH1 input

0001: COMP1 output

Others: Reserved

### 22.4.31 TIMx register map

TIMx registers are mapped as described in the table below:

**Table 125. TIM2/TIM3/TIM4 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UJFREMA	Res.	CKD [1:0]	ARPE	CMS [1:0]	DIR	OPM	URS	UDIS	CEN			
	Reset value																						0		0	0	0	0	0	0	0	0		
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T1S	MMS[2:0]	CCDS	Res.	Res.	Res.			
	Reset value																									0	0	0	0	0				
0x08	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]	Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]	Res.	SMS[2:0]	Res.	Res.	Res.		
	Reset value												0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0			0	0	0	0	0		0		0	0	0	0	0
0x10	TIMx_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.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																											0		0	0	0	0	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																											0		0	0	0	0	0
0x18	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]	Res.	Res.	OC2PE	OC2FE	CC2S [1:0]	OC1CE	OC1M [2:0]	OC1PE	OC1FE	CC1S [1:0]	Res.	Res.	Res.		
	Reset value							0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]	Res.	Res.	IC2 PSC [1:0]	CC2S [1:0]	Res.	IC1F[3:0]	IC1 PSC [1:0]	CC1S [1:0]	Res.	Res.	Res.	Res.	Res.		
Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	TIMx_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]	Res.	Res.	OC4PE	OC4FE	CC4S [1:0]	OC3CE	OC3M [2:0]	OC3PE	OC3FE	CC3S [1:0]	Res.	Res.	Res.		
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]	Res.	Res.	IC4 PSC [1:0]	CC4S [1:0]	Res.	IC3F[3:0]	IC3 PSC [1:0]	CC3S [1:0]	Res.	Res.	Res.	Res.	Res.		
Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	



Table 125. TIM2/TIM3/TIM4 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	CNT[30:16] (TIM2 only, reserved on the other timers)																CNT[15:0]																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x28	TIMx_PSC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PSC[15:0]																		
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x2C	TIMx_ARR	ARR[31:16] (TIM2 only, reserved on the other timers)																ARR[15:0]																		
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1				
0x30	Reserved																																			
0x34	TIMx_CCR1	CCR1[31:16] (TIM2 only, reserved on the other timers)																CCR1[15:0]																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x38	TIMx_CCR2	CCR2[31:16] (TIM2 only, reserved on the other timers)																CCR2[15:0]																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x3C	TIMx_CCR3	CCR3[31:16] (TIM2 only, reserved on the other timers)																CCR3[15:0]																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x40	TIMx_CCR4	CCR4[31:16] (TIM2 only, reserved on the other timers)																CCR4[15:0]																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x44	Reserved																																			
0x48	TIMx_DCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DBL[4:0]				Res	Res	Res	DBA[4:0]				
	Reset value																																0	0	0	0
0x4C	TIMx_DMAR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DMAB[15:0]																		
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x50	TIM2_OR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OCREF_CLR		
	Reset value																																	0	0	





Table 125. TIM2/TIM3/TIM4 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x50	TIM3_OR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OCREF_CLR		
	Reset value																																	0	0		
0x50	TIM4_OR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OCREF_CLR		
	Reset value																																	0	0		
0x60	TIM2_AF1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ETRSEL [3:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
	Reset value																0	0	0	0																	
0x60	TIM3_AF1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ETRSEL [3:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																0	0	0	0																	
0x60	TIM4_AF1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ETRSEL [3:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																0	0	0	0																	
0x68	TIM2_TISEL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI3SEL[3:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1SEL[3:0]
	Reset value															0	0	0	0					0	0	0	0							0	0	0	0
0x68	TIM3_TISEL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI3SEL[3:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1SEL[3:0]
	Reset value															0	0	0	0					0	0	0	0							0	0	0	0
0x68	TIM4_TISEL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI3SEL[3:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TI1SEL[3:0]
	Reset value															0	0	0	0					0	0	0	0							0	0	0	0

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

## 23 Basic timers (TIM6/TIM7)

### 23.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

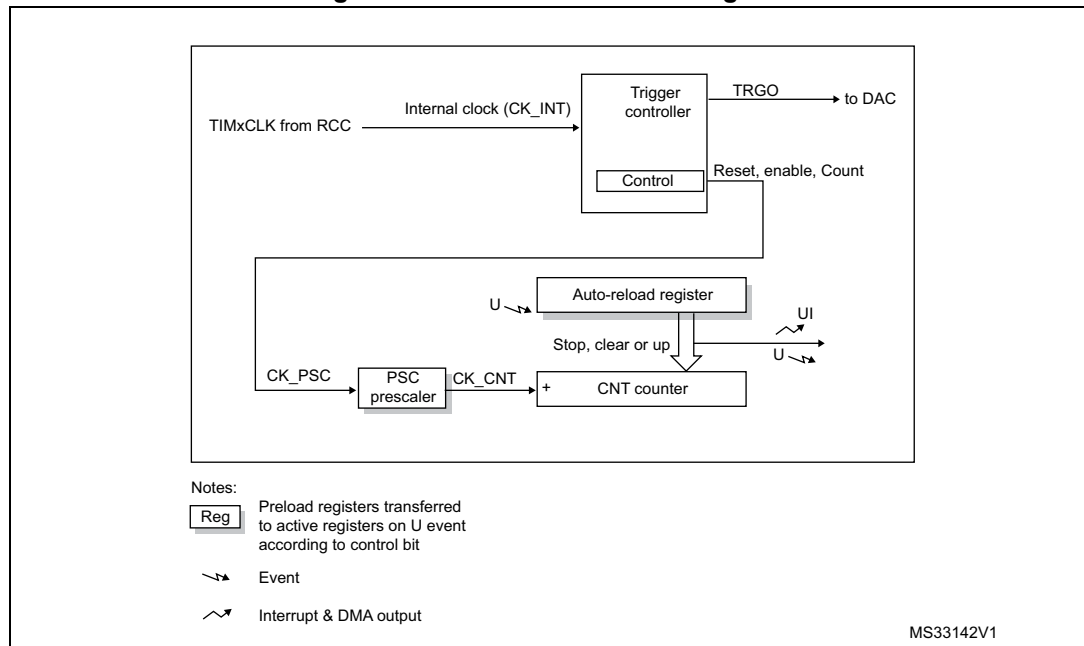
The timers are completely independent, and do not share any resources.

### 23.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 211. Basic timer block diagram



## 23.3 TIM6/TIM7 functional description

### 23.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC)
- Auto-Reload Register (TIMx\_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx\_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx\_CR1 register is set.

Note that the actual counter enable signal CNT\_EN is set 1 clock cycle after CEN.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as the TIMx\_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 212* and *Figure 213* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 212. Counter timing diagram with prescaler division change from 1 to 2

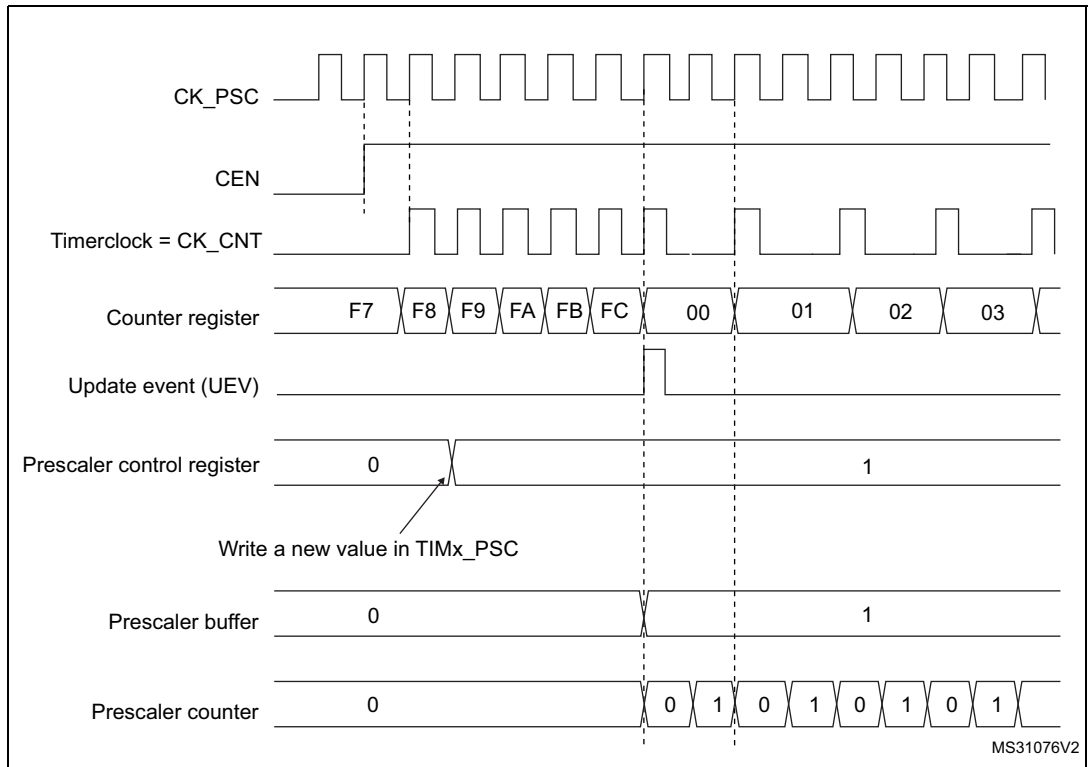
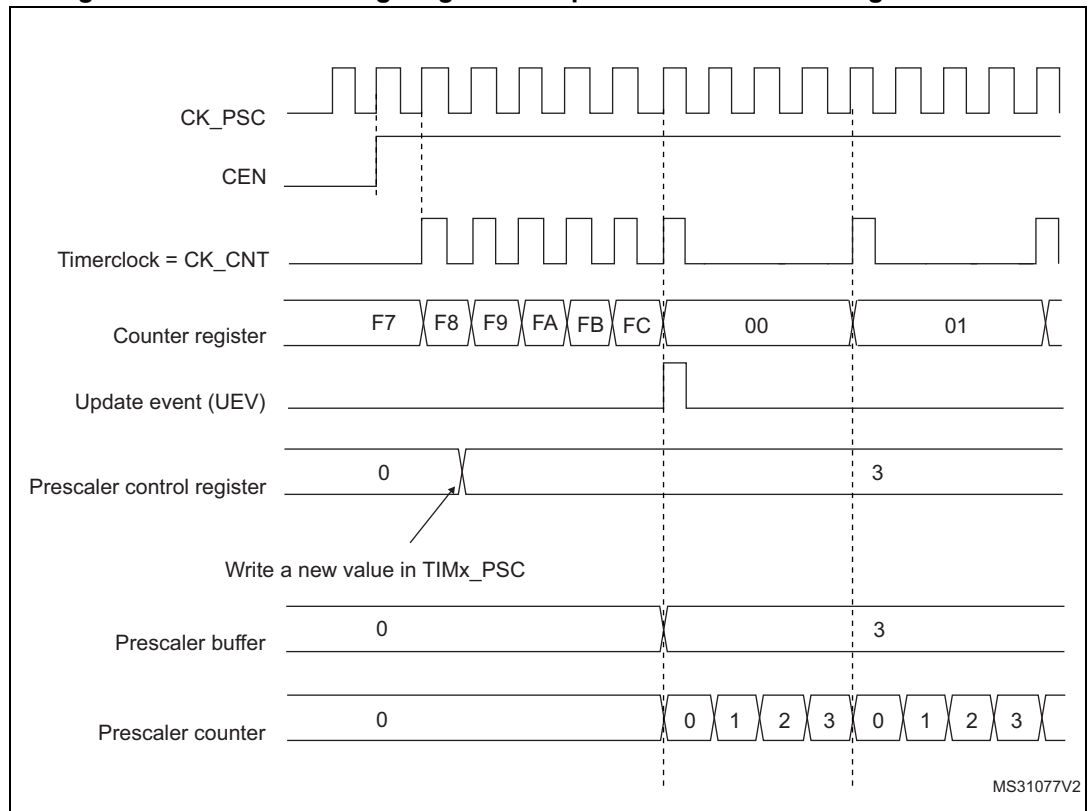


Figure 213. Counter timing diagram with prescaler division change from 1 to 4



### 23.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been cleared, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx\_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx\_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx\_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR = 0x36.

**Figure 214. Counter timing diagram, internal clock divided by 1**

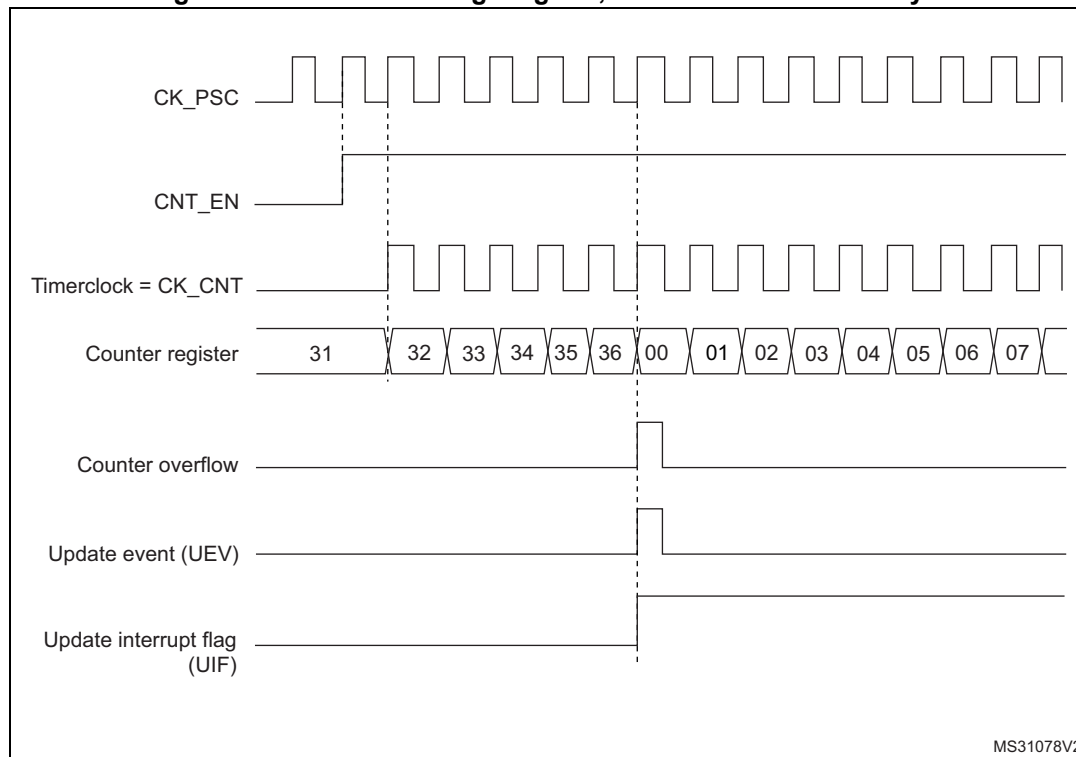


Figure 215. Counter timing diagram, internal clock divided by 2

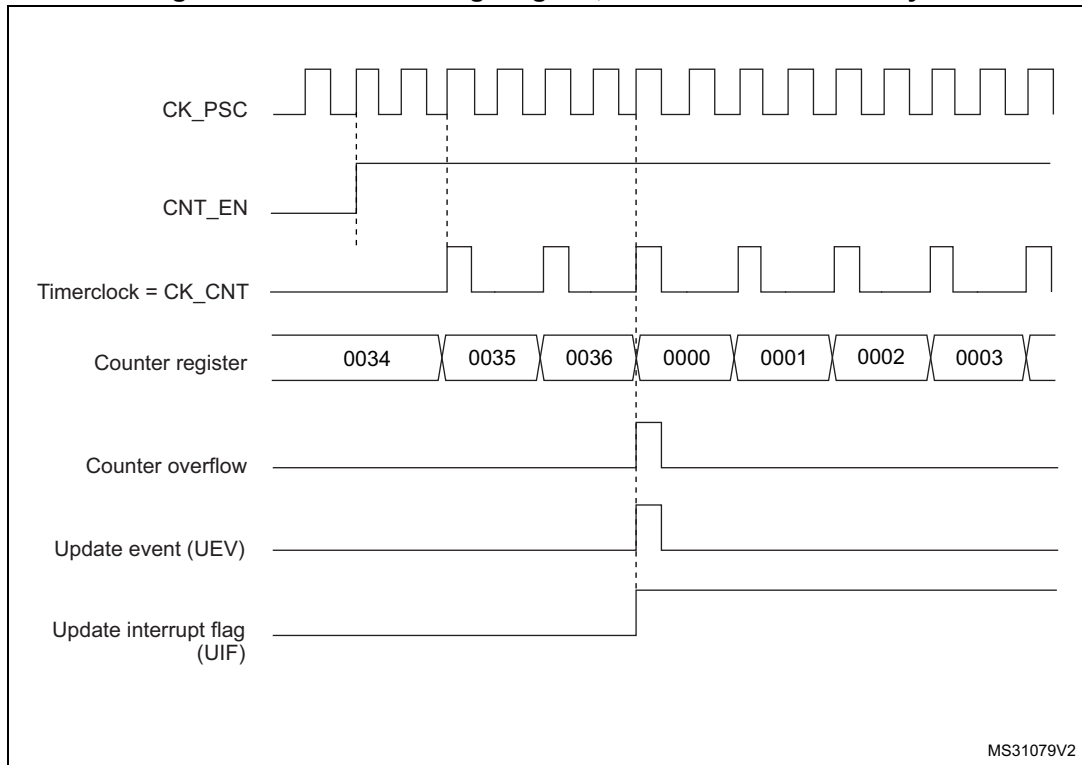


Figure 216. Counter timing diagram, internal clock divided by 4

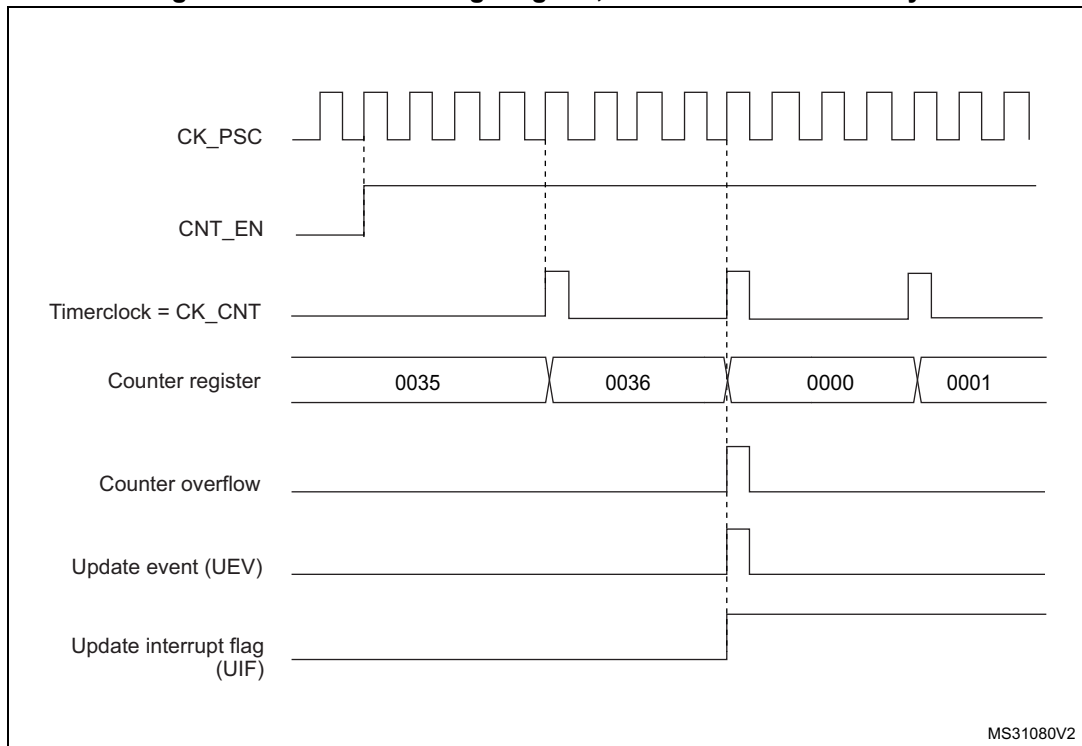
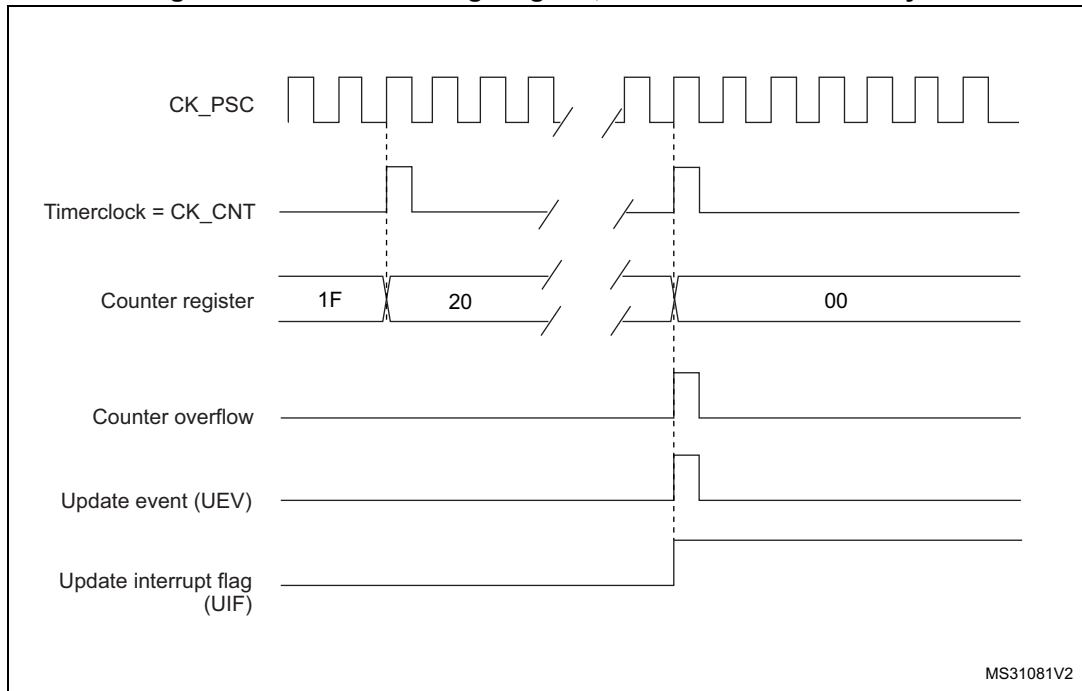
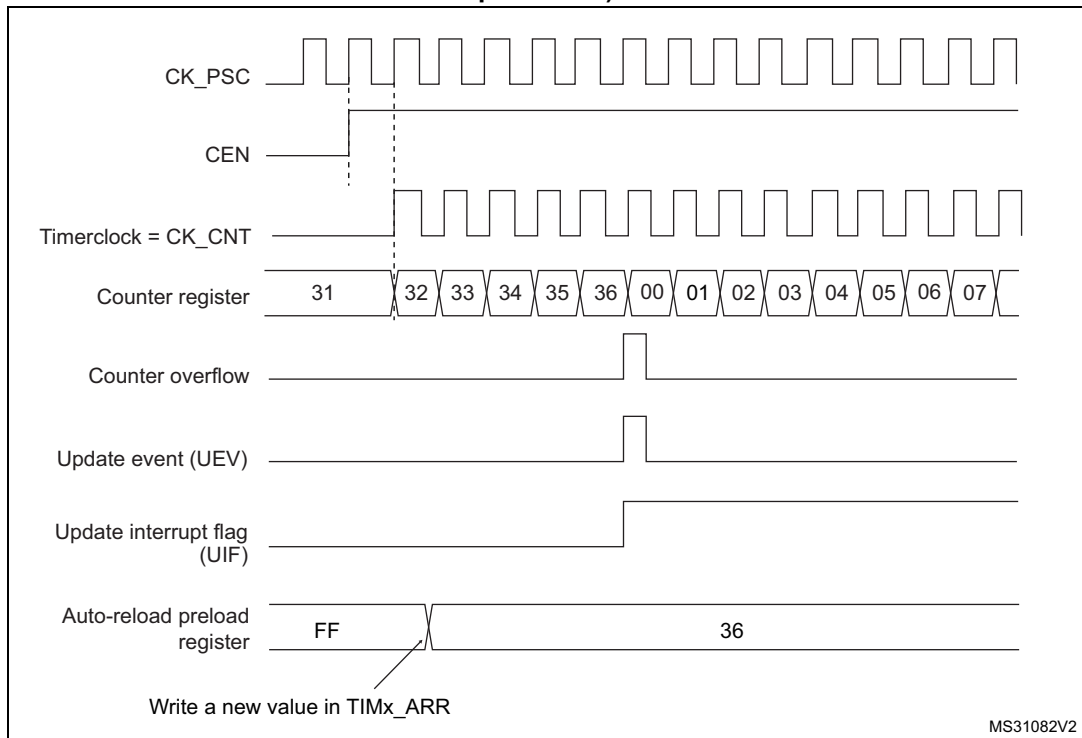


Figure 217. Counter timing diagram, internal clock divided by N



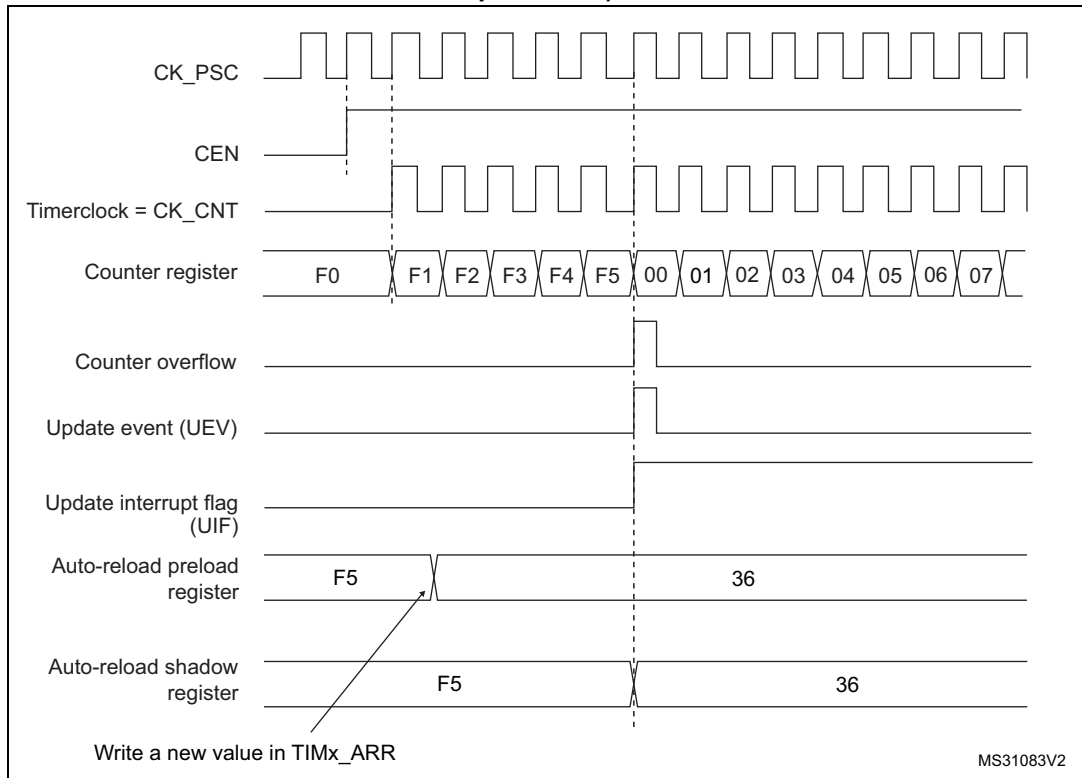
MS31081V2

Figure 218. Counter timing diagram, update event when ARPE = 0 (TIMx\_ARR not preloaded)



MS31082V2

**Figure 219. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 23.3.3 UIF bit remapping

The IUFREMAP bit in the TIMx\_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

### 23.3.4 Clock source

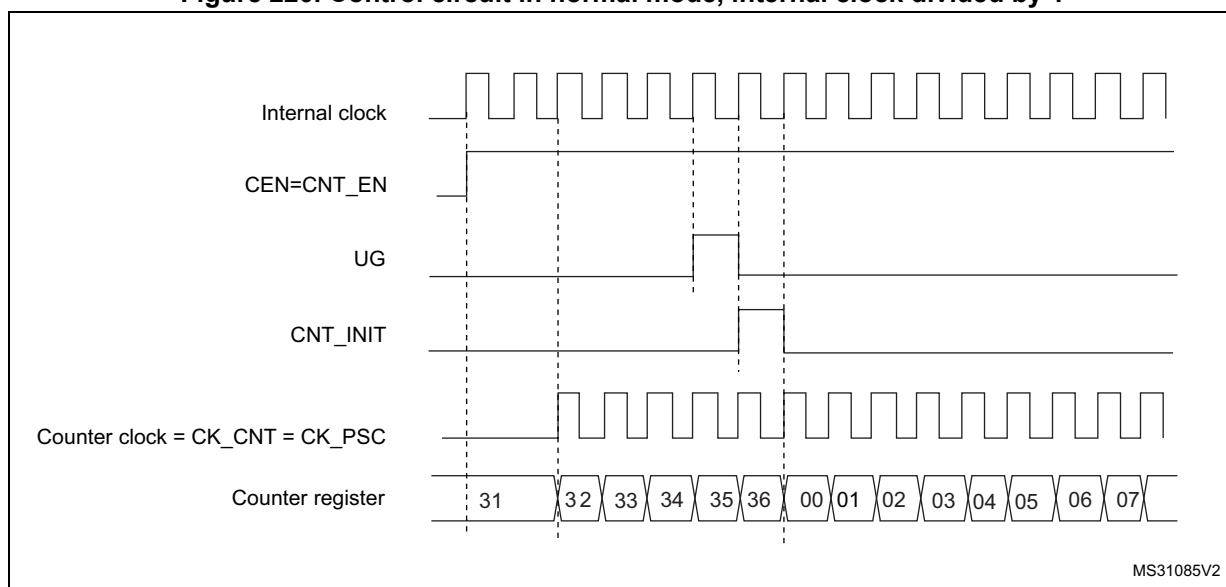
The counter clock is provided by the Internal clock (CK\_INT) source.

The CEN (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

*Figure 220* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.



Figure 220. Control circuit in normal mode, internal clock divided by 1



### 23.3.5 Debug mode

When the microcontroller enters the debug mode (Cortex<sup>®</sup>-M0+ core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG\_TIMx\_STOP configuration bit in the DBG module. For more details, refer to [Section 40.9.2: Debug support for timers, watchdog and I<sup>2</sup>C](#).

## 23.4 TIM6/TIM7 registers

Refer to [Section 1.2 on page 55](#) for a list of abbreviations used in register descriptions.

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

### 23.4.1 TIMx control register 1 (TIMx\_CR1)(x = 6 to 7)

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.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				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.

Bits 10:8 Reserved, must be kept at reset value.

- Bit 7 **ARPE**: Auto-reload preload enable  
0: TIMx\_ARR register is not buffered.  
1: TIMx\_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

- Bit 3 **OPM**: One-pulse mode  
0: Counter is not stopped at update event  
1: Counter stops counting at the next update event (clearing the CEN bit).

- Bit 2 **URS**: Update request source  
This bit is set and cleared by software to select the UEV event sources.  
0: Any of the following events generates an update interrupt or DMA request if enabled.  
These events can be:
- Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

- Bit 1 **UDIS**: Update disable  
This bit is set and cleared by software to enable/disable UEV event generation.  
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- Buffered registers are then loaded with their preload values.  
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

- Bit 0 **CEN**: Counter enable  
0: Counter disabled  
1: Counter enabled

*Note: Gated mode can work only if the CEN bit has been previously set by software.  
However trigger mode can set the CEN bit automatically by hardware.*

CEN is cleared automatically in one-pulse mode, when an update event occurs.

### 23.4.2 TIMx control register 2 (TIMx\_CR2)(x = 6 to 7)

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.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

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

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx\_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT\_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx\_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bits 3:0 Reserved, must be kept at reset value.

### 23.4.3 TIMx DMA/Interrupt enable register (TIMx\_DIER)(x = 6 to 7)

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.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

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

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.

1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.

1: Update interrupt enabled.

### 23.4.4 TIMx status register (TIMx\_SR)(x = 6 to 7)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx\_CR1 register.
- When CNT is reinitialized by software using the UG bit in the TIMx\_EGR register, if URS = 0 and UDIS = 0 in the TIMx\_CR1 register.

### 23.4.5 TIMx event generation register (TIMx\_EGR)(x = 6 to 7)

Address offset: 0x14

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

### 23.4.6 TIMx counter (TIMx\_CNT)(x = 6 to 7)

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

### 23.4.7 TIMx prescaler (TIMx\_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value  
 The counter clock frequency CK\_CNT is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .  
 PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 23.4.8 TIMx auto-reload register (TIMx\_ARR)(x = 6 to 7)

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]**: Prescaler value  
 ARR is the value to be loaded into the actual auto-reload register.  
 Refer to [Section 23.3.1: Time-base unit on page 695](#) for more details about ARR update and behavior.  
 The counter is blocked while the auto-reload value is null.

### 23.4.9 TIMx register map

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

**Table 126. TIMx register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UJFREMA	Res.	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN												
	Reset value																						0				0				0	0	0	0												
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.													
	Reset value																											0	0	0																
0x08	Reserved																																													
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	UIE										
0x10	TIMx_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.	Res.	Res.												
	Reset value																																			UIF										
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG											
	Reset value																																			0										
0x18-0x20	Reserved																																													
0x24	TIMx_CNT	UJFCPY or Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]												
	Reset value	0																																			0 0									
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]										
	Reset value																																					0 0								
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]									
	Reset value																																						1 1							

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 24 General-purpose timers (TIM14)

### 24.1 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.

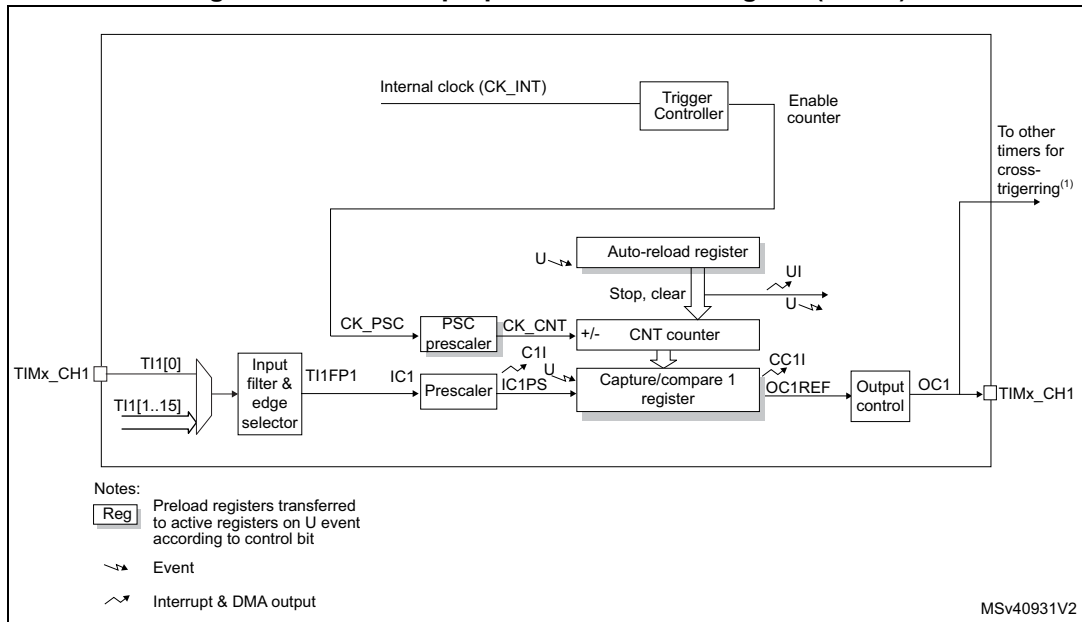
### 24.2 TIM14 main features

#### 24.2.1 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 221. General-purpose timer block diagram (TIM14)



1. This signal can be used as trigger for some slave timers, see [Section 24.3.11: Using timer output as trigger for other timers \(TIM14\)](#).



## 24.3 TIM14 functional description

### 24.3.1 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 222](#) and [Figure 223](#) give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 222. Counter timing diagram with prescaler division change from 1 to 2

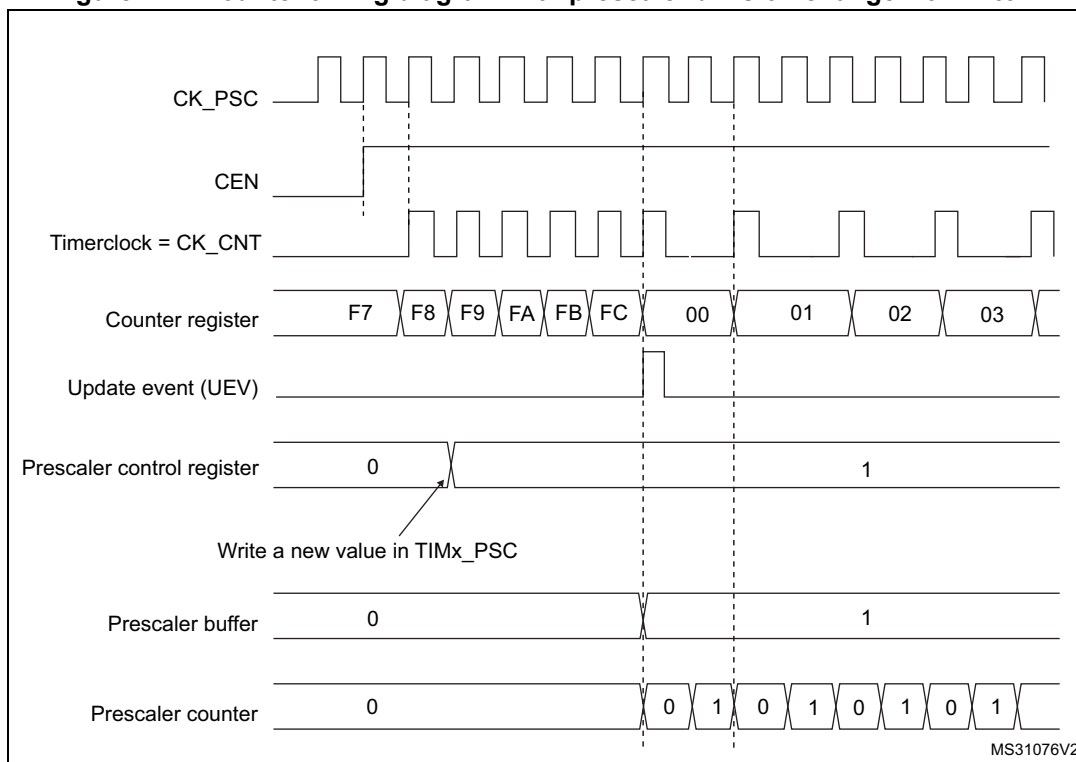
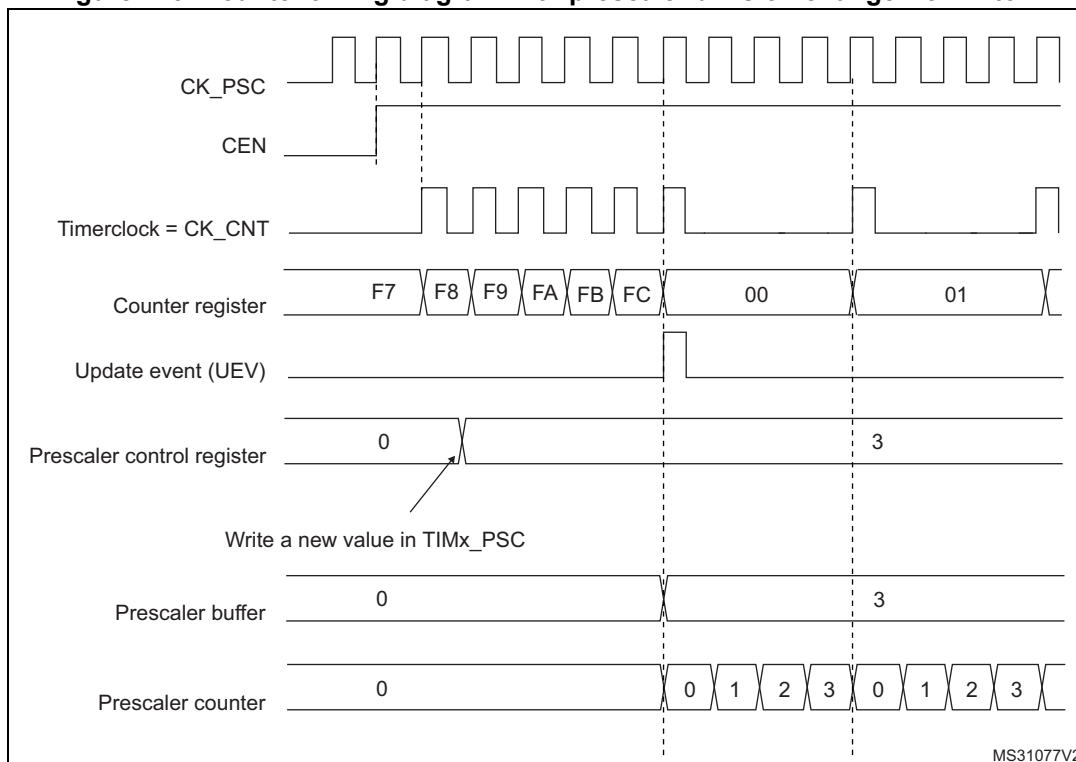


Figure 223. Counter timing diagram with prescaler division change from 1 to 4



### 24.3.2 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx\_EGR register (by software) also generates an update event.

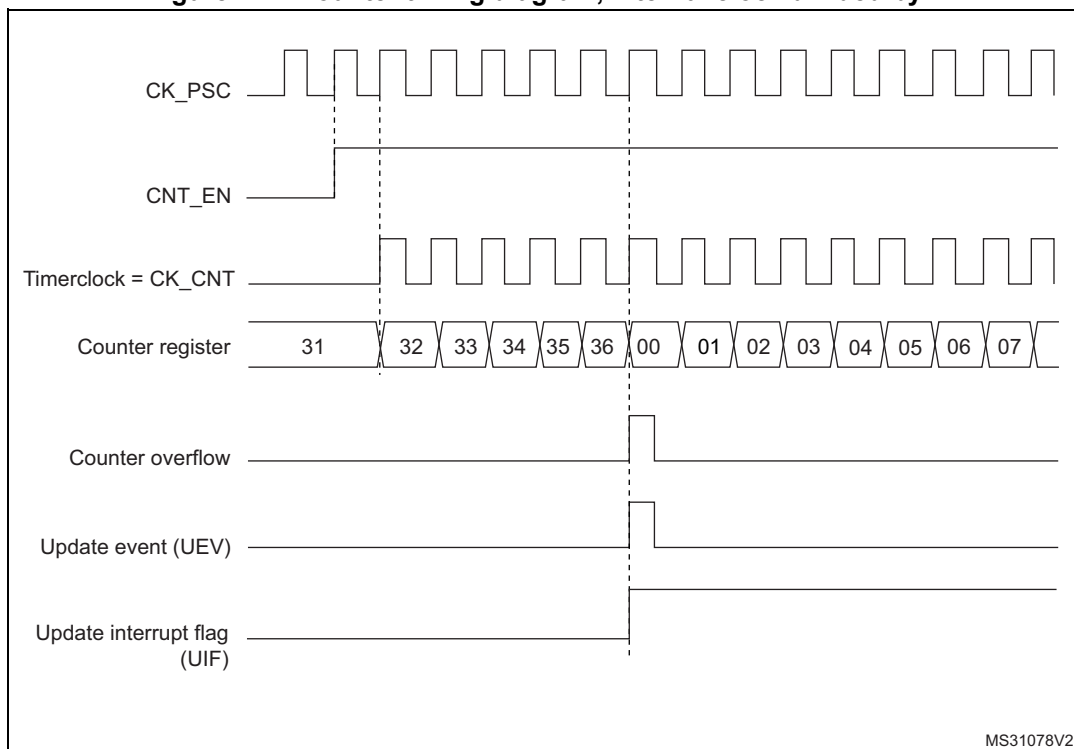
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as 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 224. Counter timing diagram, internal clock divided by 1**



MS31078V2

Figure 225. Counter timing diagram, internal clock divided by 2

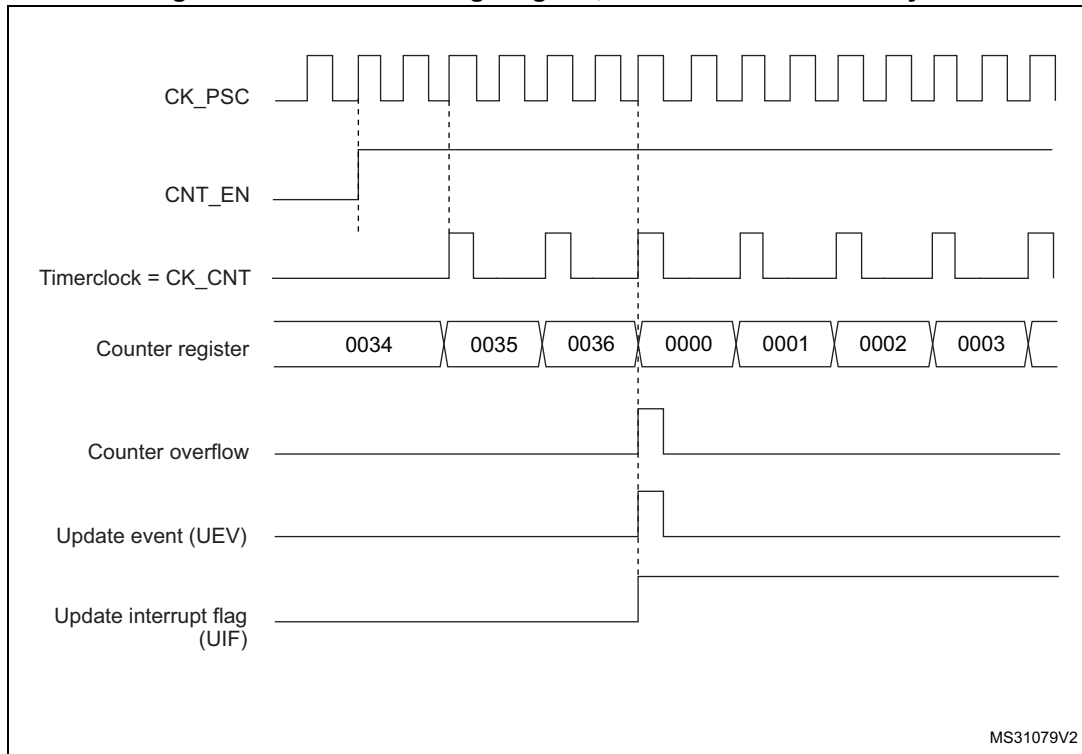


Figure 226. Counter timing diagram, internal clock divided by 4

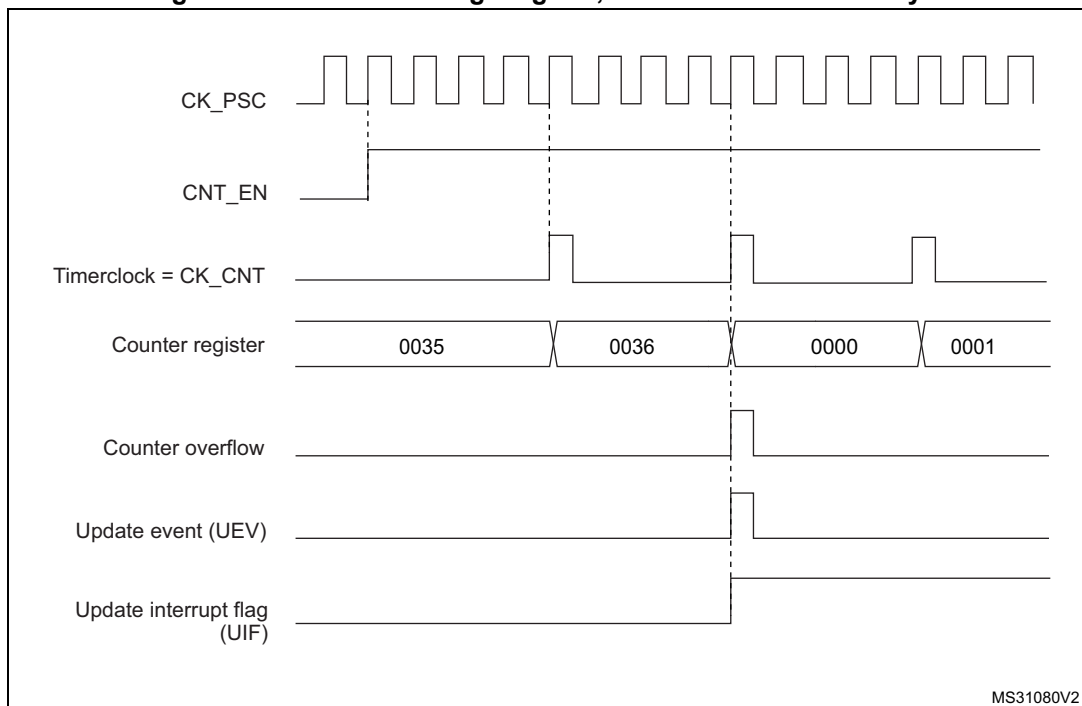


Figure 227. Counter timing diagram, internal clock divided by N

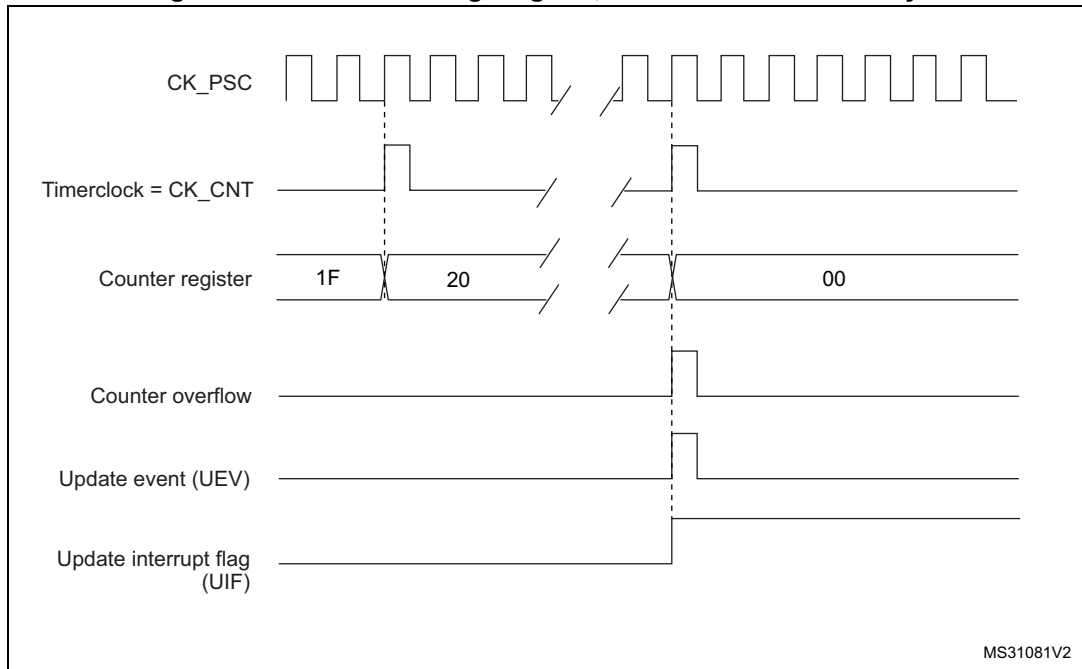
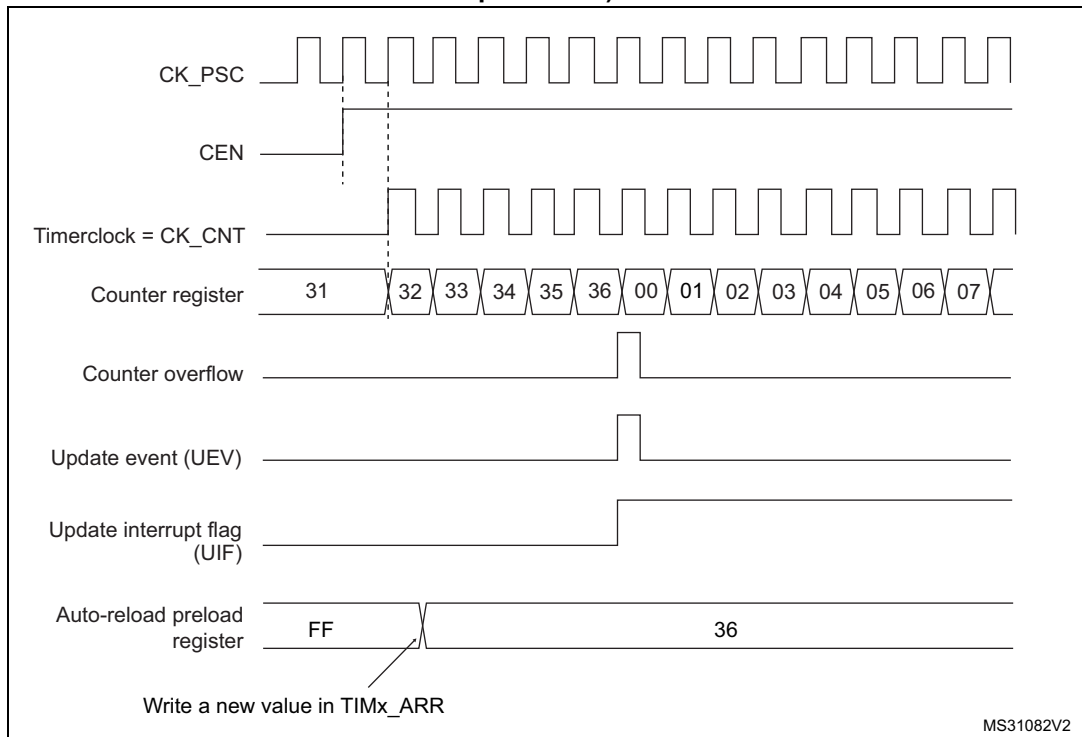
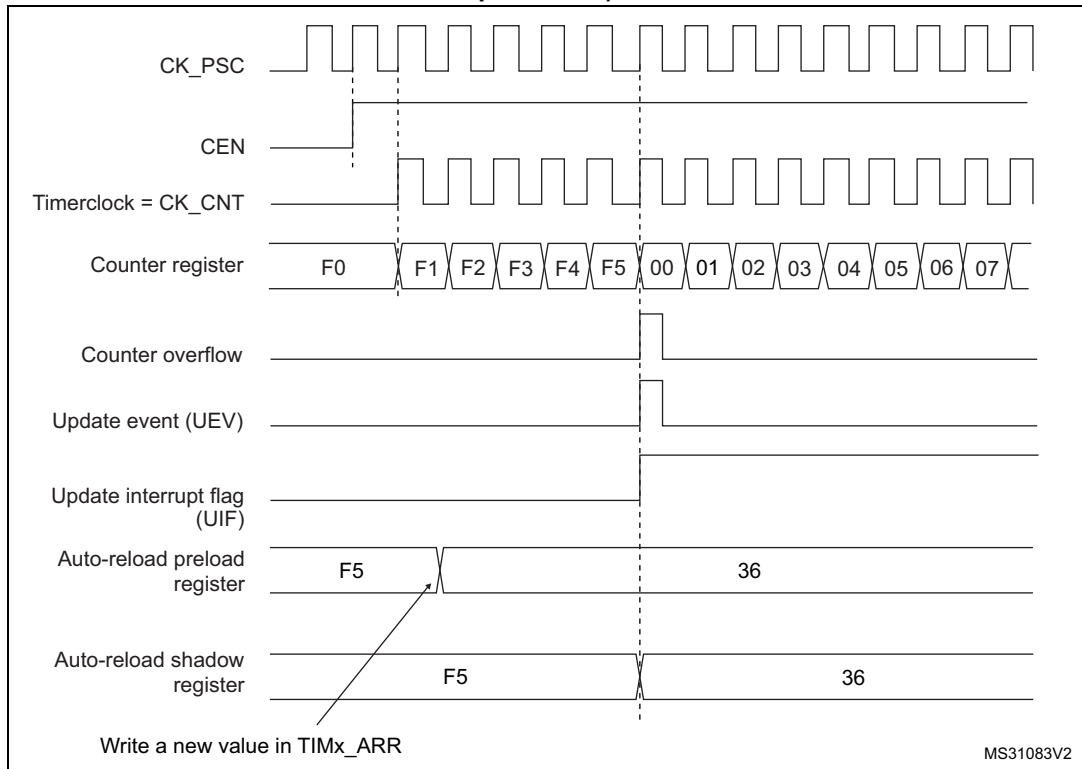


Figure 228. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)



**Figure 229. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 24.3.3 Clock selection

The counter clock can be provided by the following clock sources:

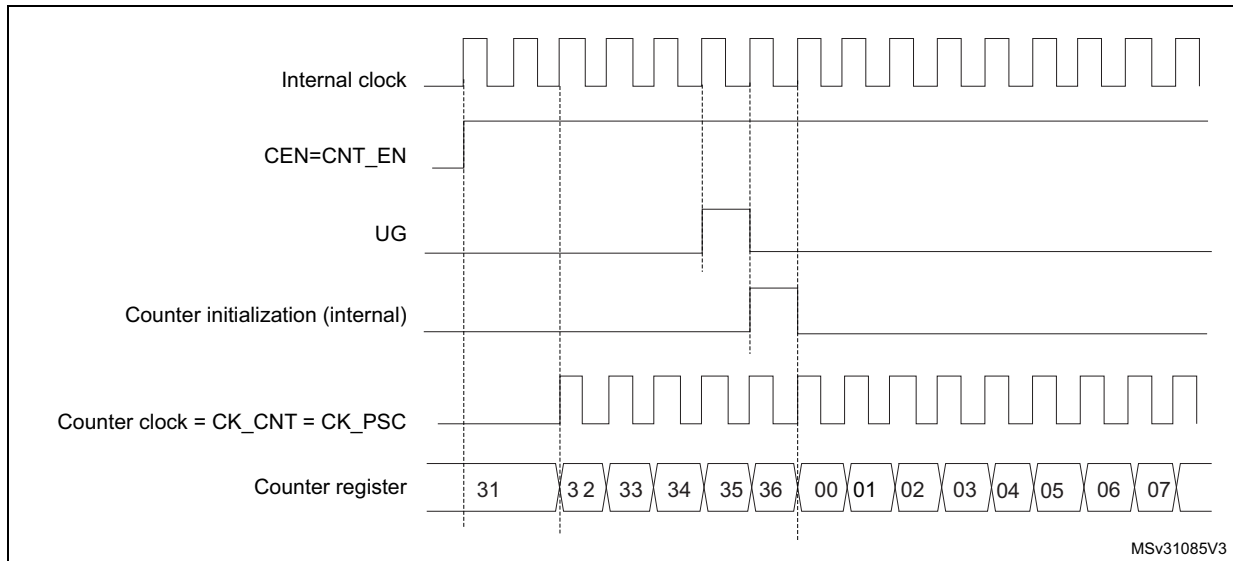
- Internal clock (CK\_INT)

#### Internal clock source (CK\_INT)

The internal clock source is the default clock source for TIM14.

*Figure 230* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 230. Control circuit in normal mode, internal clock divided by 1



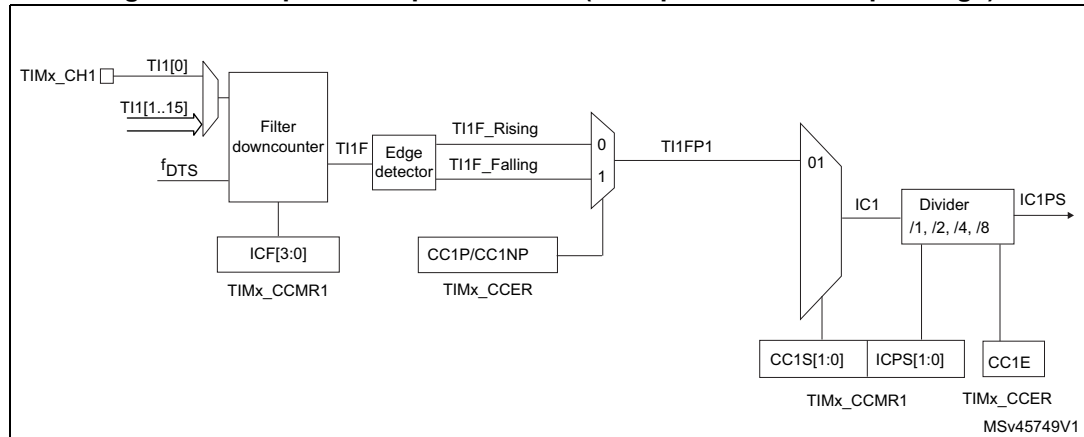
### 24.3.4 Capture/compare channels

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 231 to Figure 233 give an overview of one capture/compare channel.

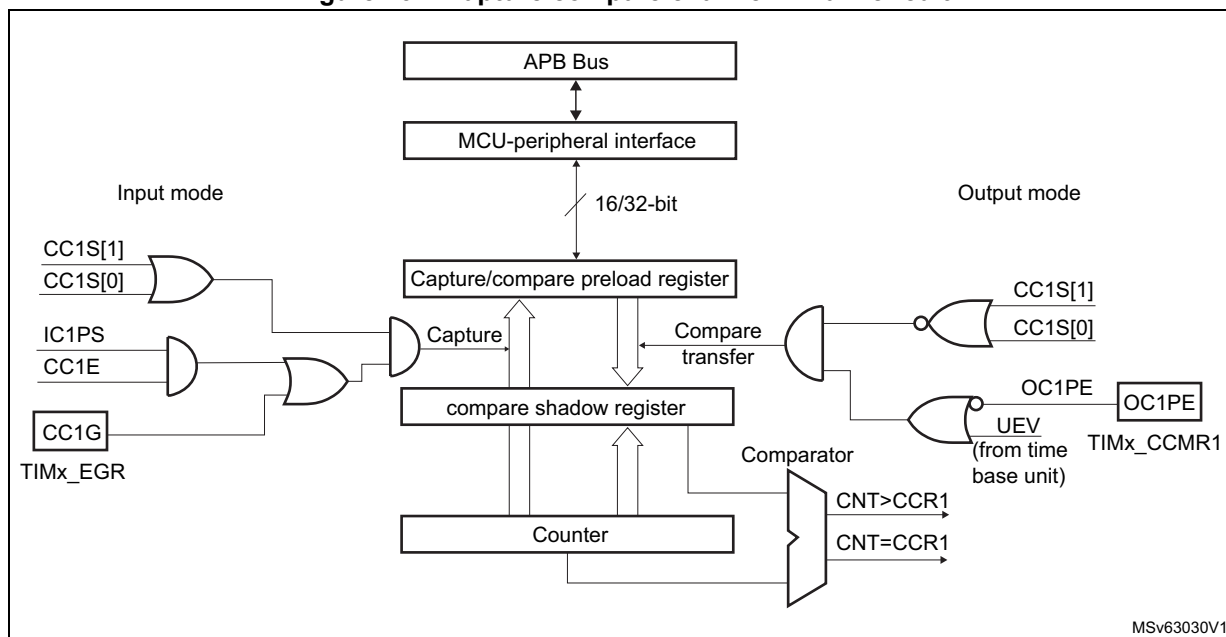
The input stage samples the corresponding Tix input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TixFPx) which can be used as the capture command. It is prescaled before the capture register (ICxPS).

Figure 231. Capture/compare channel (example: channel 1 input stage)



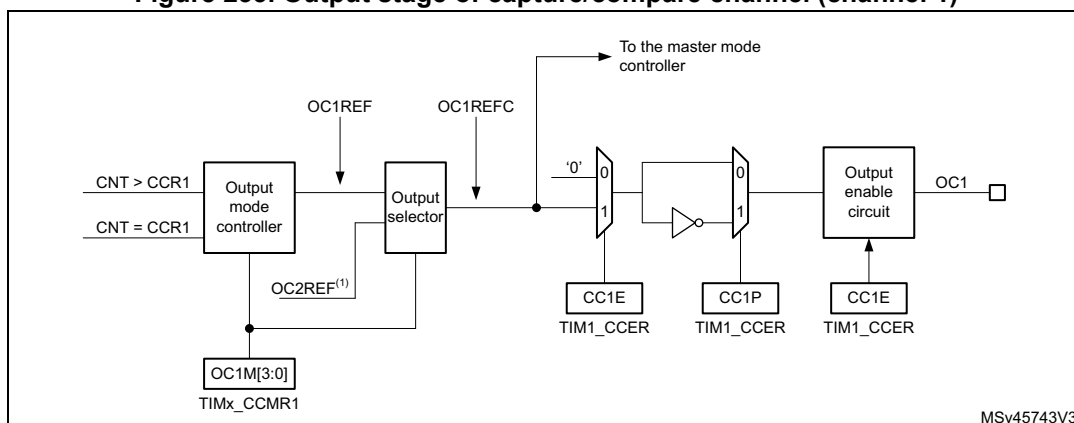
The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 232. Capture/compare channel 1 main circuit



MSv63030V1

Figure 233. Output stage of capture/compare channel (channel 1)



MSv45743V3

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.

### 24.3.5 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.

### 24.3.6 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.

### 24.3.7 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.

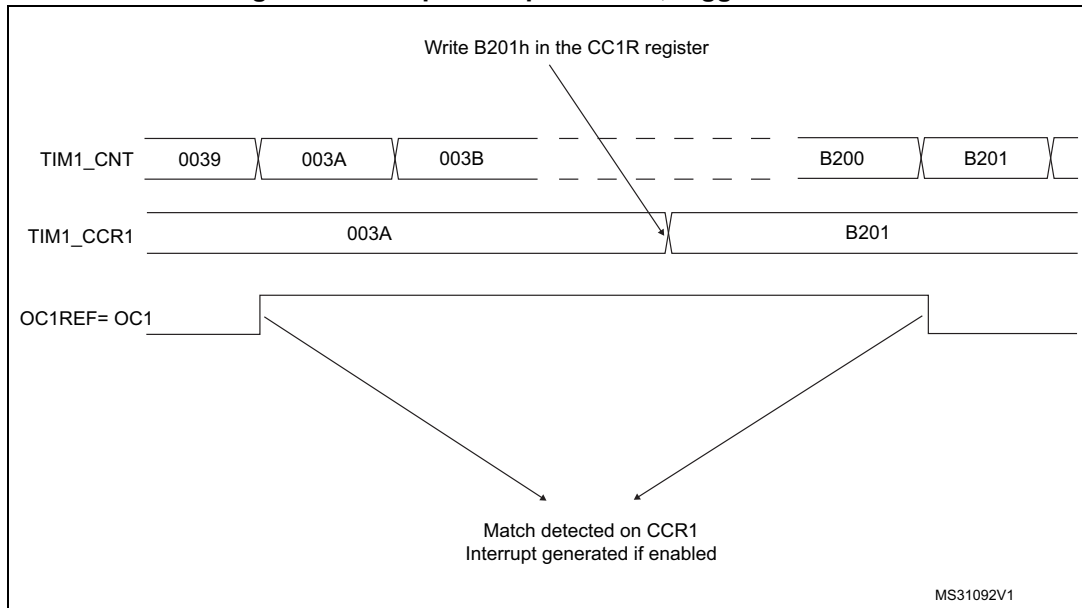
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 234](#).

Figure 234. Output compare mode, toggle on OC1.



### 24.3.8 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

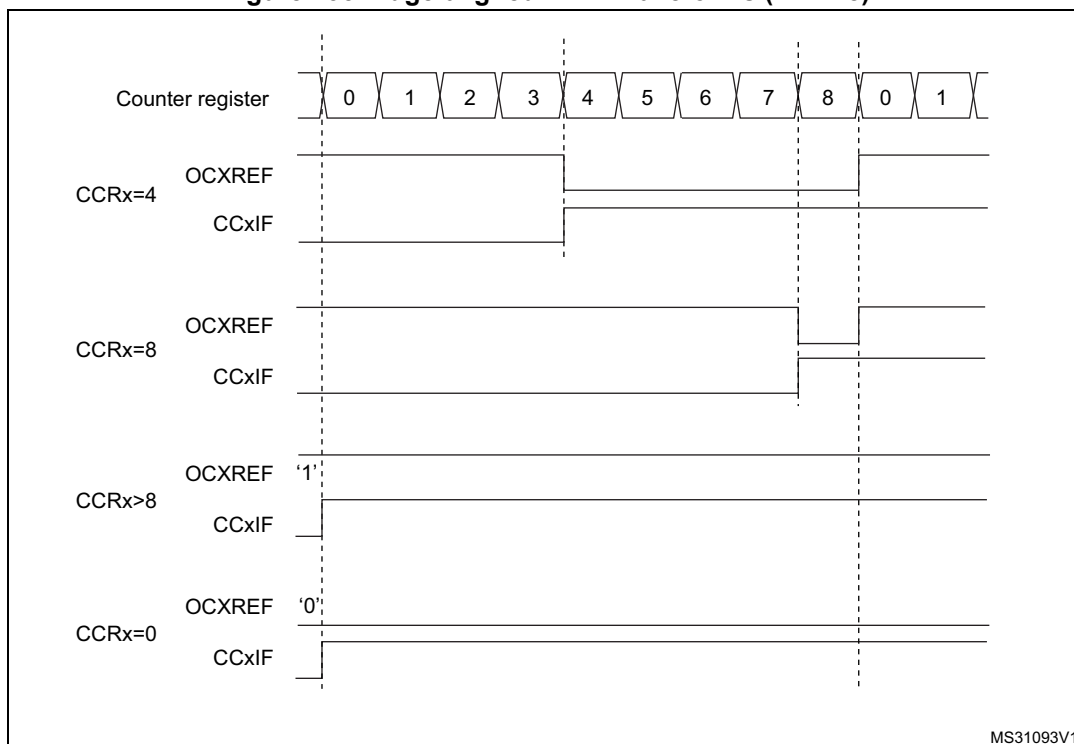
The OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx\_CCER register. Refer to the TIMx\_CCERx register description for more details.

In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $TIMx\_CNT \leq TIMx\_CCRx$ .

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as  $TIMx\_CNT < TIMx\_CCRx$  else it becomes low. If the compare value in TIMx\_CCRx is greater than the auto-reload value (in TIMx\_ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'. [Figure 235](#) shows some edge-aligned PWM waveforms in an example where  $TIMx\_ARR=8$ .

Figure 235. Edge-aligned PWM waveforms (ARR=8)



MS31093V1

### 24.3.9 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:

$$CNT < CCRx = ARR \text{ (in particular, } 0 < CCRx)$$

### 24.3.10 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.

### 24.3.11 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.

### 24.3.12 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 40.9.2: Debug support for timers, watchdog and I<sup>2</sup>C](#).

## 24.4 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).

### 24.4.1 TIM14 control register 1 (TIM14\_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	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.*

### 24.4.2 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

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

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 24.4.3 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.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0



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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

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.

### 24.4.4 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w



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

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

### 24.4.5 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.	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.	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).

#### 24.4.6 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.	Res.	Res.	Res.	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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).*

### 24.4.7 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

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

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.  
 CC1 channel configured as output: CC1NP must be kept cleared.  
 CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

- 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 TlxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TlxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TlxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 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 127. 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.

### 24.4.8 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r/w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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



### 24.4.9 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”).

### 24.4.10 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 24.3.1: Time-base unit on page 709](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 24.4.11 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).

### 24.4.12 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T1SEL[3:0]			
												rw	rw	rw	rw

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

Bits 3:0 **T1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM14\_CH1 input

0001: RTC CLK

0010: HSE/32

0011: MCO

0100: MCO2<sup>(1)</sup>

Others: Reserved

1. Available on STM32G0B1xx and STM32G0C1xx salestypes only.

### 24.4.13 TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

**Table 128. 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	<b>TIMx_CR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN	
	Reset value																						0		0	0	0				0	0	0	0
0x04 to 0x08	Reserved	Res.																																
0x0C	<b>TIMx_DIER</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																	0



Table 128. TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																			
0x10	TIMx_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.	CC1F	UJF																		
	Reset value																									0								0	0																	
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG																
	Reset value																																		0	0																
0x18	TIMx_CCMR1 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]				OC1PE	OC1FE	CC1S [1:0]																		
	Reset value																0											0	0	0	0	0	0	0	0	0																
	TIMx_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]	CC1S [1:0]																			
	Reset value																											0	0	0	0	0	0	0	0	0																
0x1C	Reserved	Res.																																																		
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1P	CC1E															
	Reset value																																			0	0	0														
0x24	TIMx_CNT	UIFCPY		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]																
	Reset value	0																																			0	0	0	0	0	0	0	0	0	0	0	0				
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]														
	Reset value																																					0	0	0	0	0	0	0	0	0	0	0	0			
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]													
	Reset value																																								0	0	0	0	0	0	0	0	0	0		
0x30	Reserved	Res.																																																		
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]								
	Reset value																																							0	0	0	0	0	0	0	0	0	0	0	0	
0x38 to 0x64	Reserved	Res.																																																		
0x68	TIM14_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T1SEL[3:0]					
	Reset value																																																	0	0	0

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.





## 25 General-purpose timers (TIM15/TIM16/TIM17)

### 25.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in [Section 25.4.23: Timer synchronization \(TIM15\)](#).

### 25.2 TIM15 main features

TIM15 includes the following features:

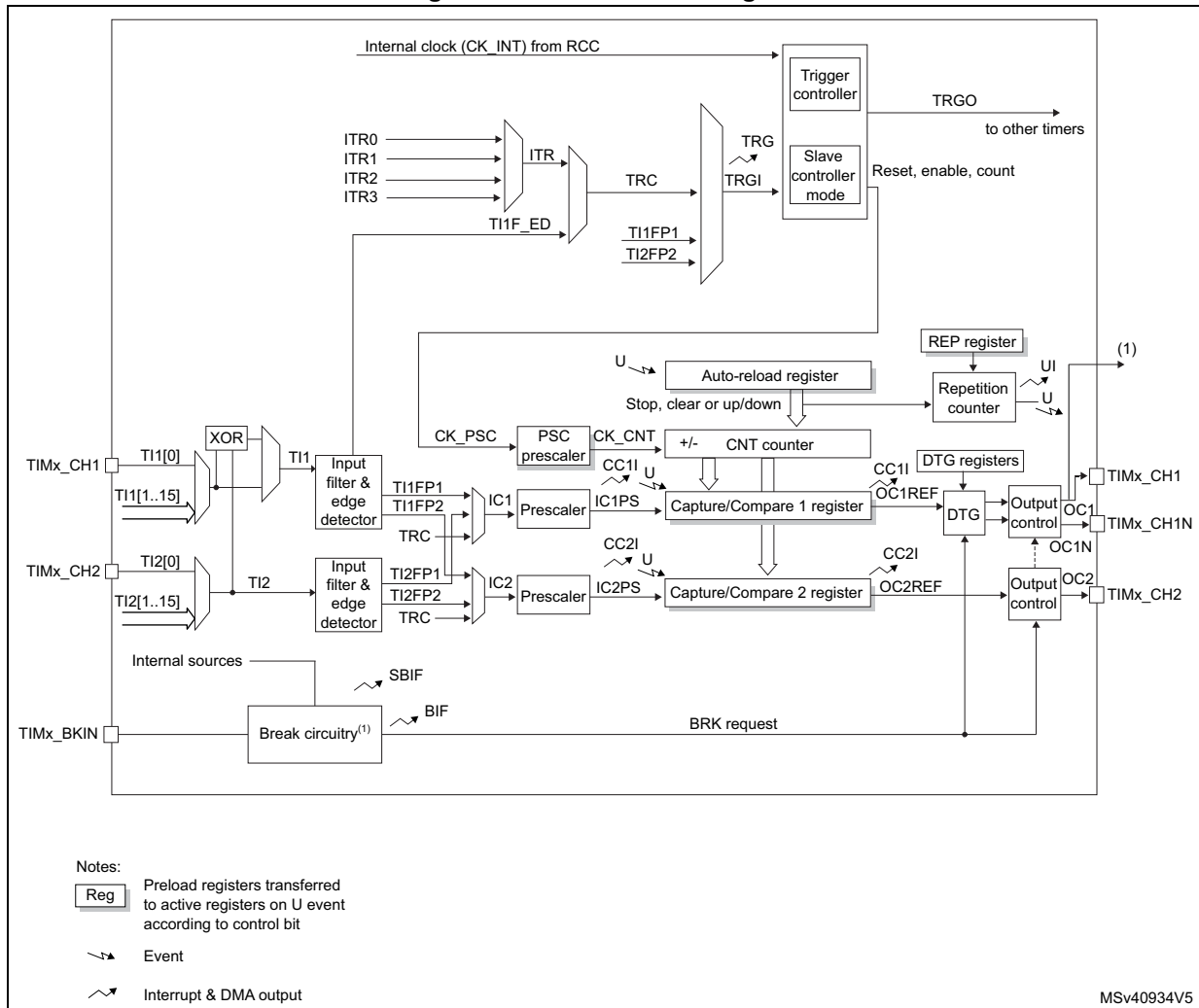
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

## 25.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

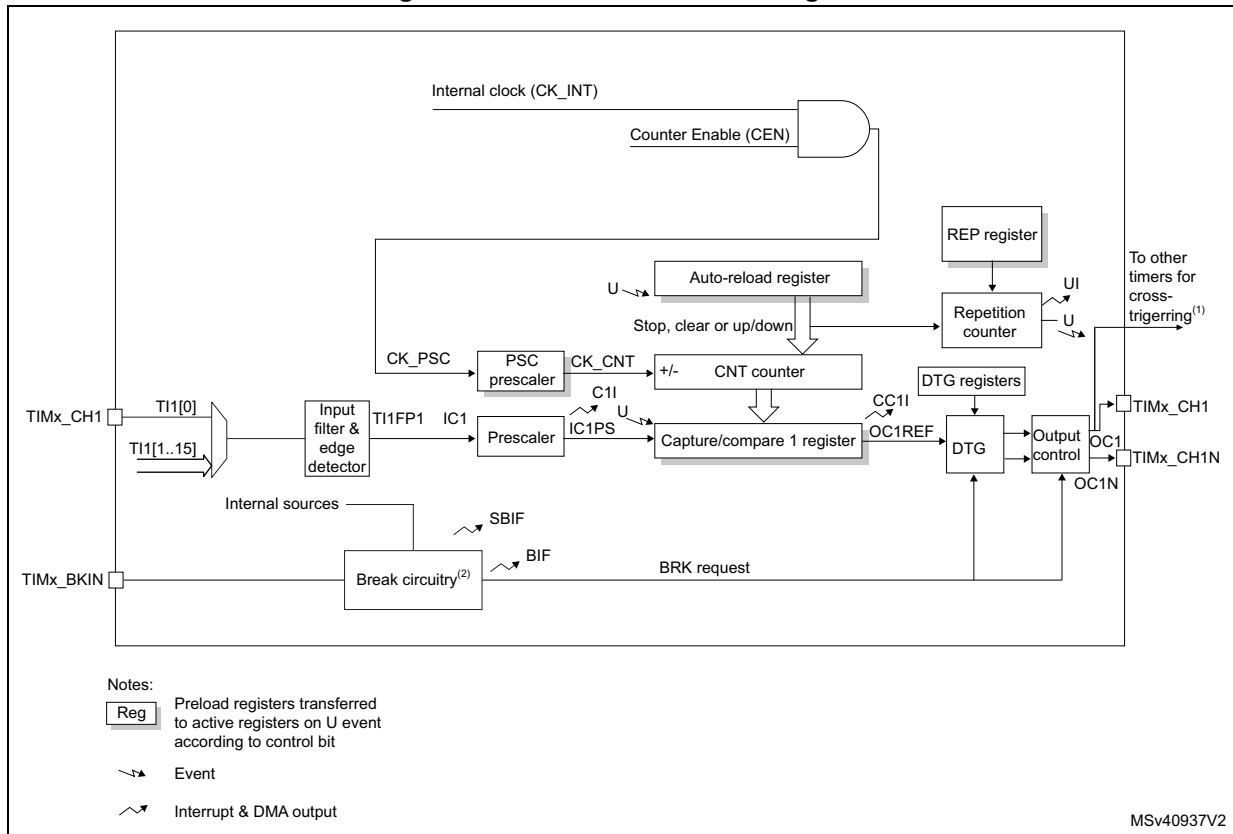
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 236. TIM15 block diagram



- The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 5.2.9: Clock security system \(CSS\)](#)
  - A PVD output
  - SRAM parity error signal
  - Cortex<sup>®</sup>-M0+ LOCKUP (Hardfault) output
  - COMP output

Figure 237. TIM16/TIM17 block diagram



- This signal can be used as trigger for some slave timer, see [Section 25.4.24: Using timer output as trigger for other timers \(TIM16/TIM17\)](#).
- The internal break event source can be:
  - A clock failure event generated by CSS. For further information on the CSS, refer to [Section 5.2.9: Clock security system \(CSS\)](#)
  - A PVD output
  - SRAM parity error signal
  - Cortex<sup>®</sup>-M0+ LOCKUP (Hardfault) output
  - COMP output

## 25.4 TIM15/TIM16/TIM17 functional description

### 25.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Auto-reload register (TIMx\_ARR)
- Repetition counter register (TIMx\_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx\_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx\_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIMx\_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx\_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx\_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 238* and *Figure 239* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 238. Counter timing diagram with prescaler division change from 1 to 2

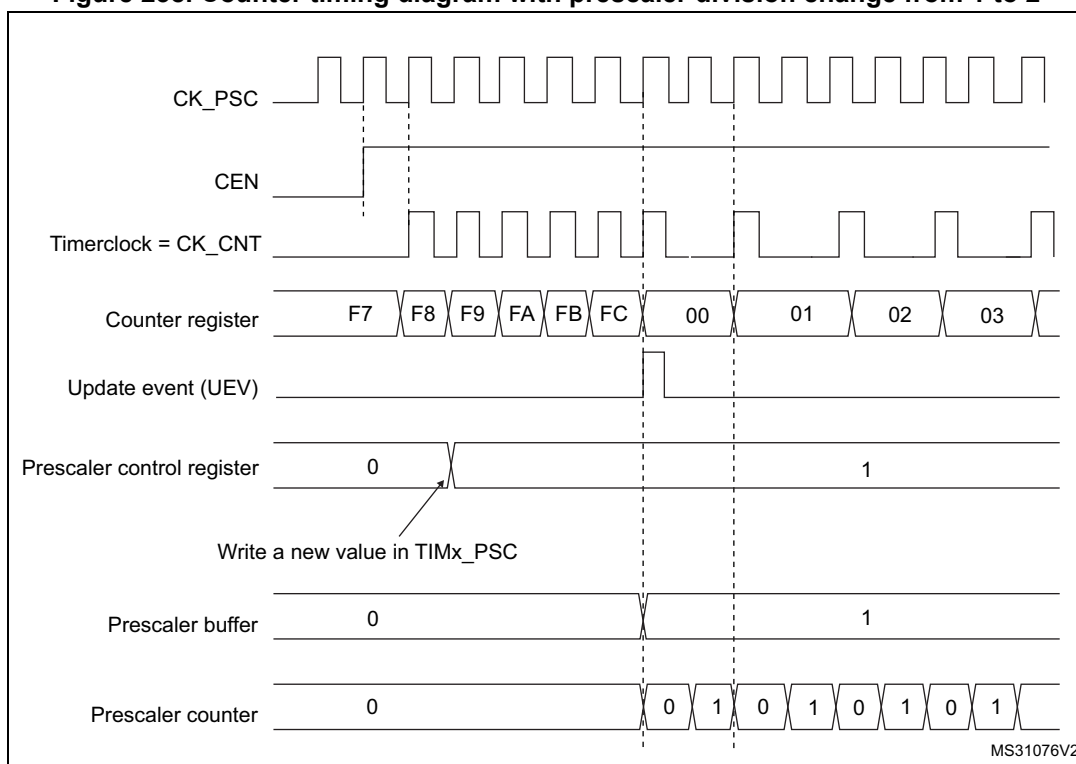
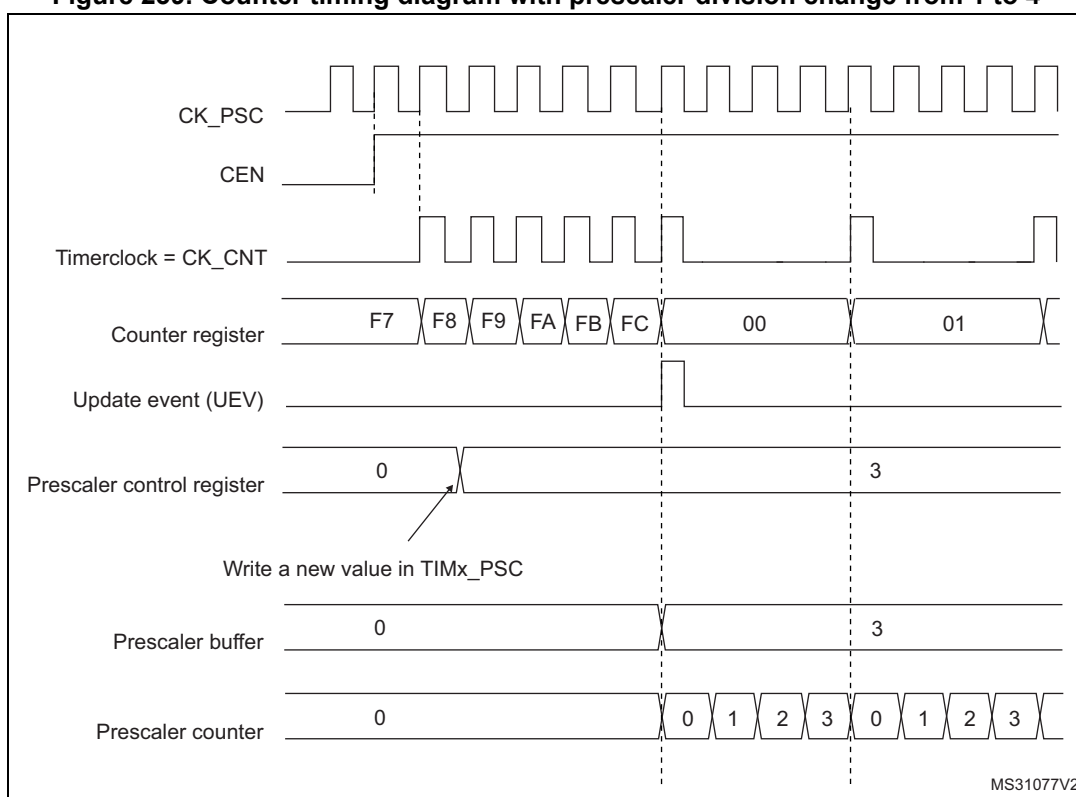


Figure 239. Counter timing diagram with prescaler division change from 1 to 4



## 25.4.2 Counter modes

### Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx\_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx\_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx\_EGR register (by software or by using the slave mode controller) also generates an update event.

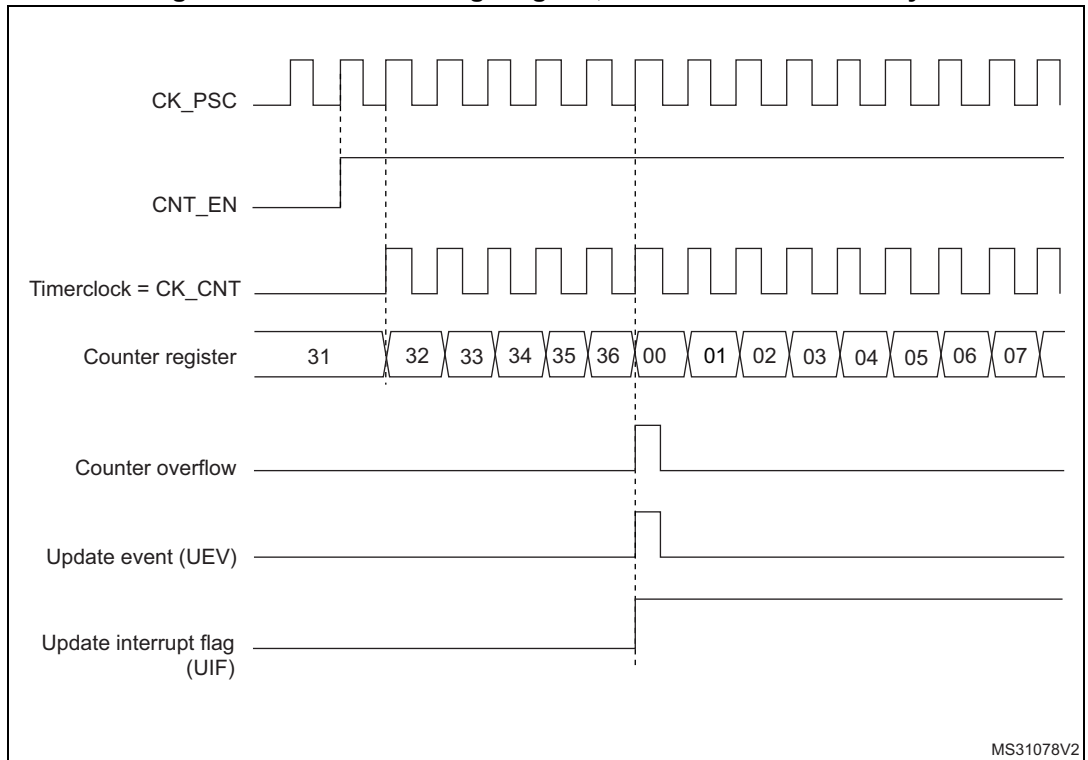
The UEV event can be disabled by software by setting the UDIS bit in the TIMx\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx\_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx\_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx\_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx\_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx\_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx\_ARR=0x36.

**Figure 240. Counter timing diagram, internal clock divided by 1**



**Figure 241. Counter timing diagram, internal clock divided by 2**

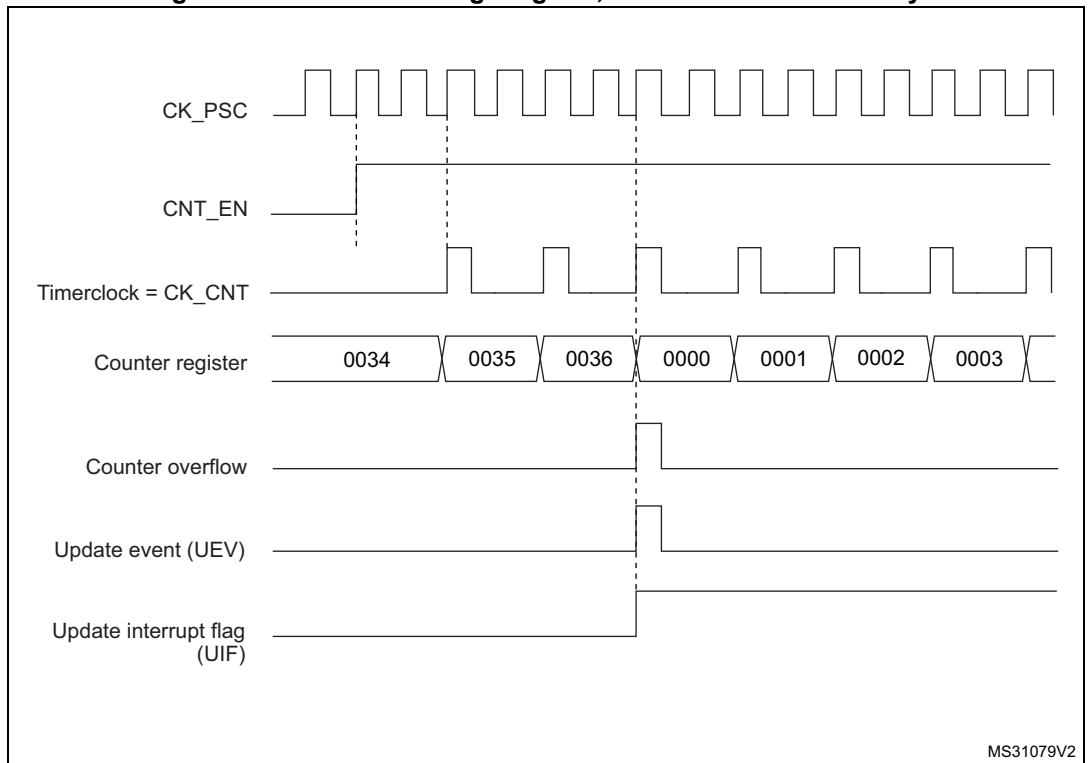
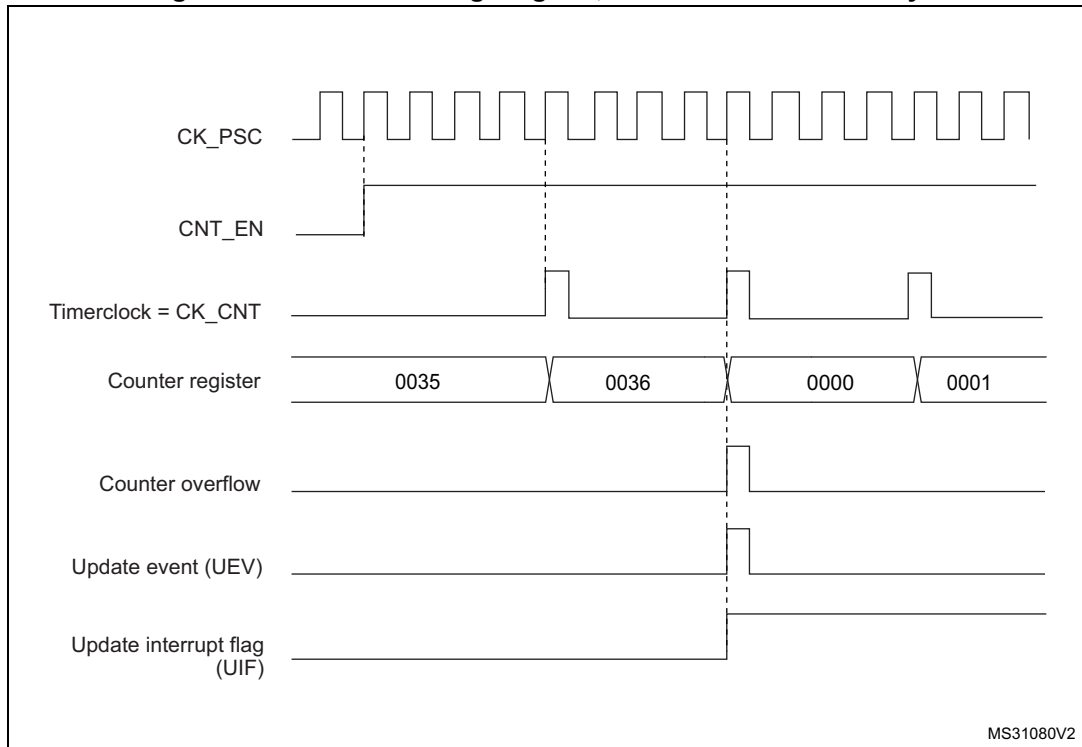


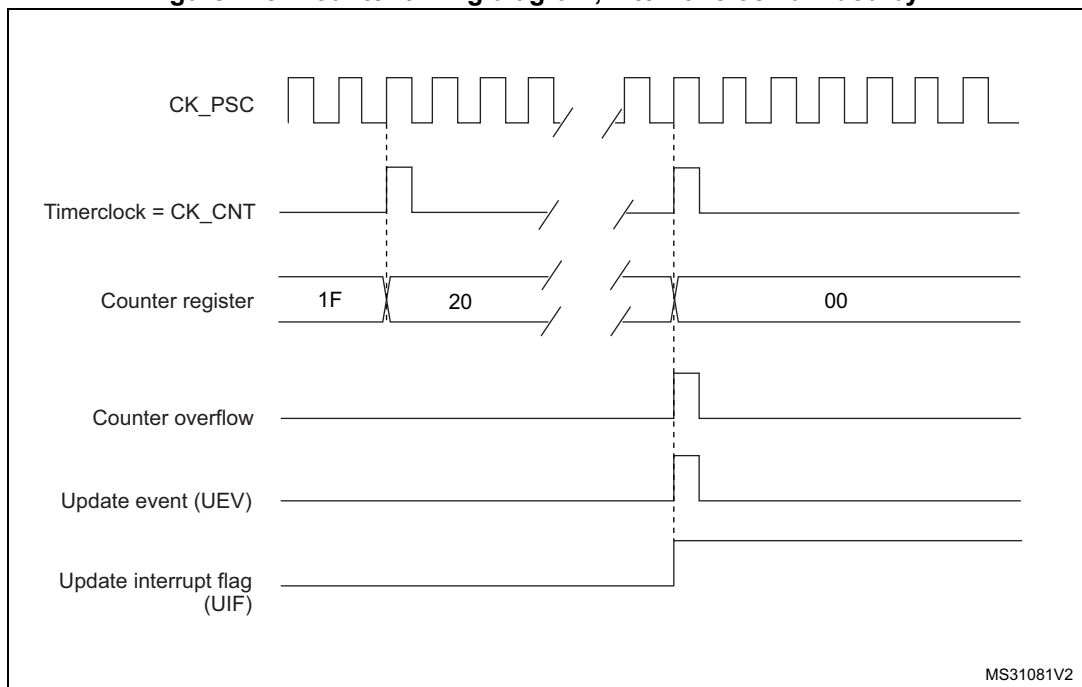


Figure 242. Counter timing diagram, internal clock divided by 4



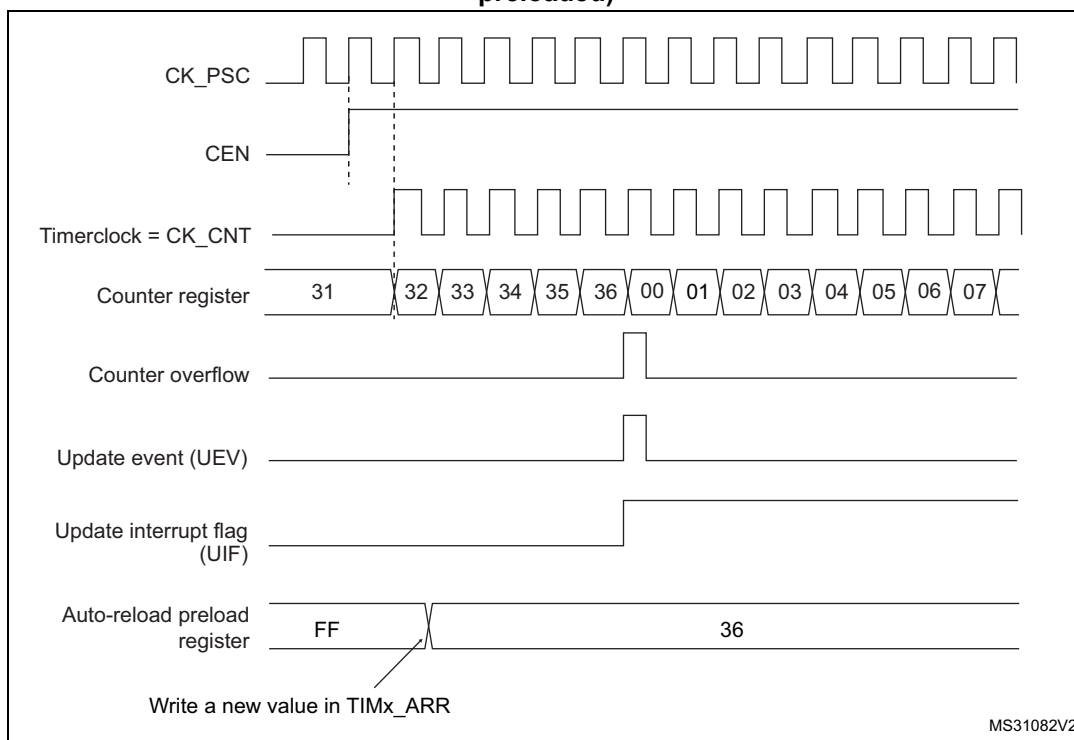
MS31080V2

Figure 243. Counter timing diagram, internal clock divided by N

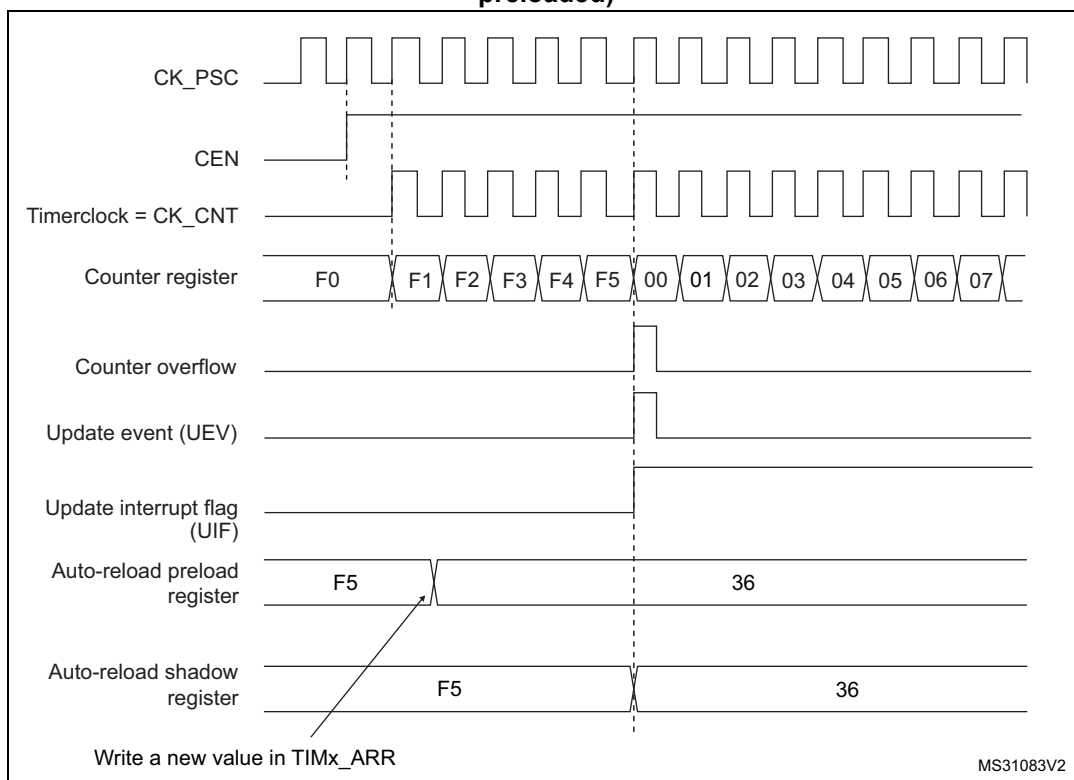


MS31081V2

**Figure 244. Counter timing diagram, update event when ARPE=0 (TIMx\_ARR not preloaded)**



**Figure 245. Counter timing diagram, update event when ARPE=1 (TIMx\_ARR preloaded)**



### 25.4.3 Repetition counter

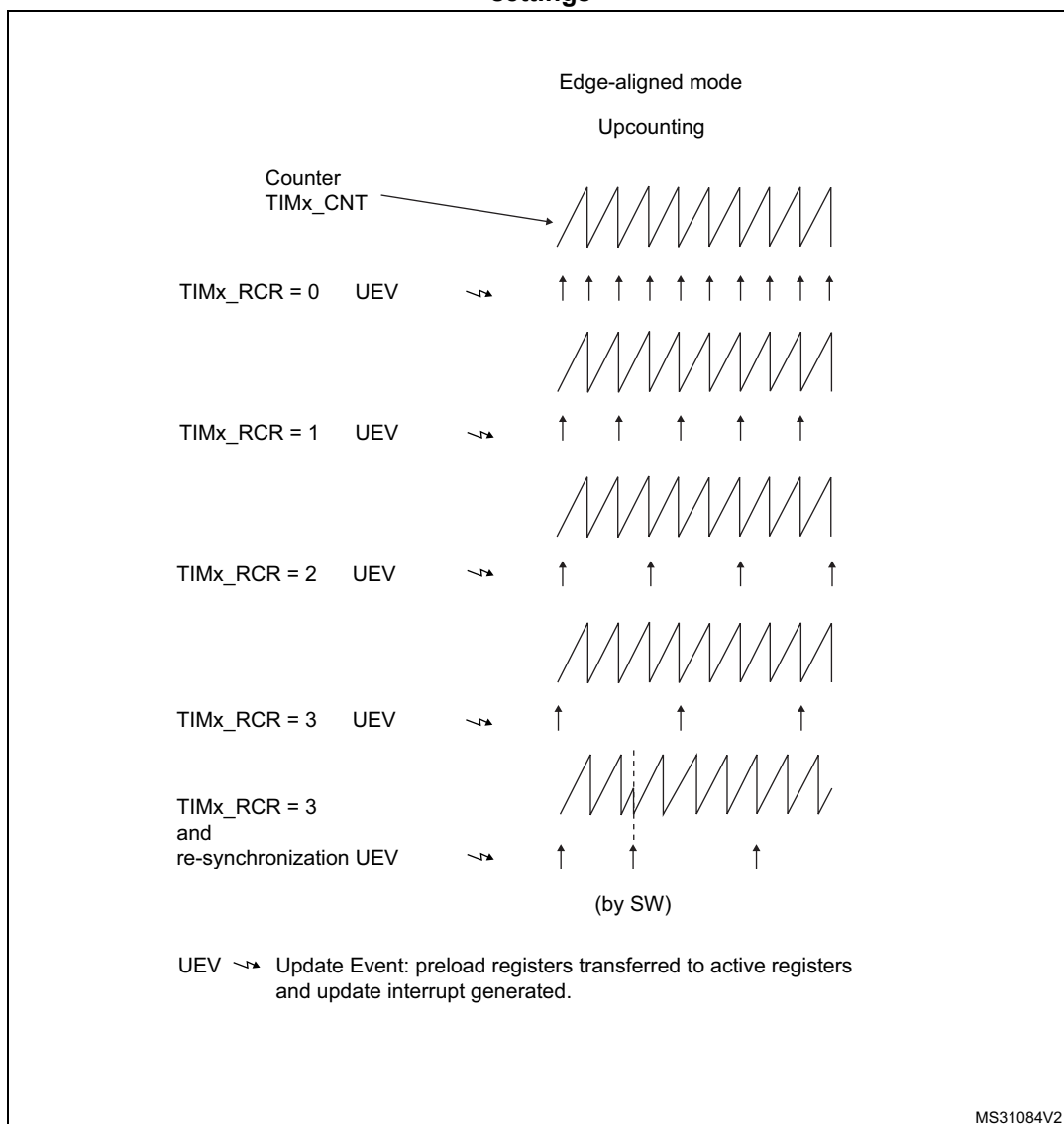
*Section 25.4.1: Time-base unit* describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx\_ARR auto-reload register, TIMx\_PSC prescaler register, but also TIMx\_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx\_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx\_RCR register value (refer to *Figure 246*). 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 246. Update rate examples depending on mode and TIMx\_RCR register settings



### 25.4.4 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK\_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to [Using one timer as prescaler for another timer on page 656](#) for more details.

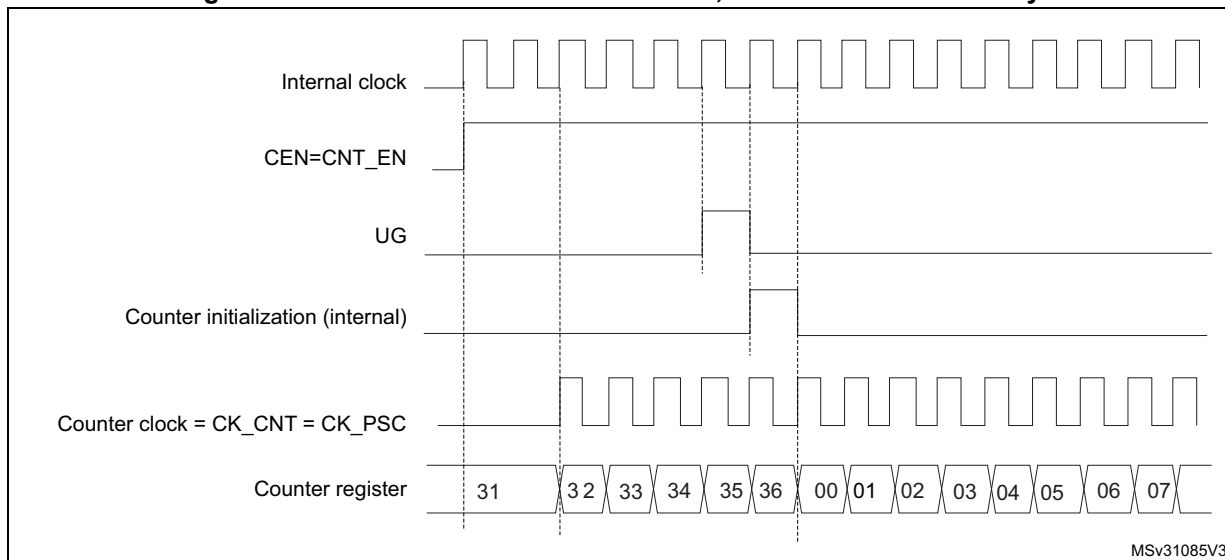
#### Internal clock source (CK\_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx\_CR1 register) and UG bits (in the TIMx\_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK\_INT.

Figure 247 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

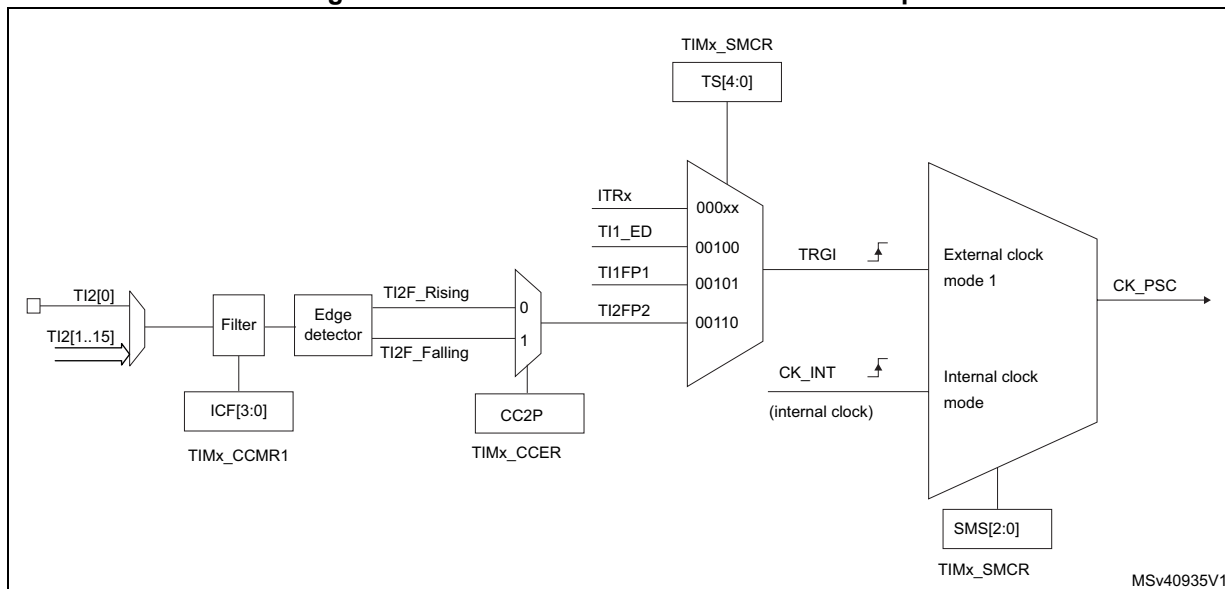
Figure 247. Control circuit in normal mode, internal clock divided by 1



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 248. 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:

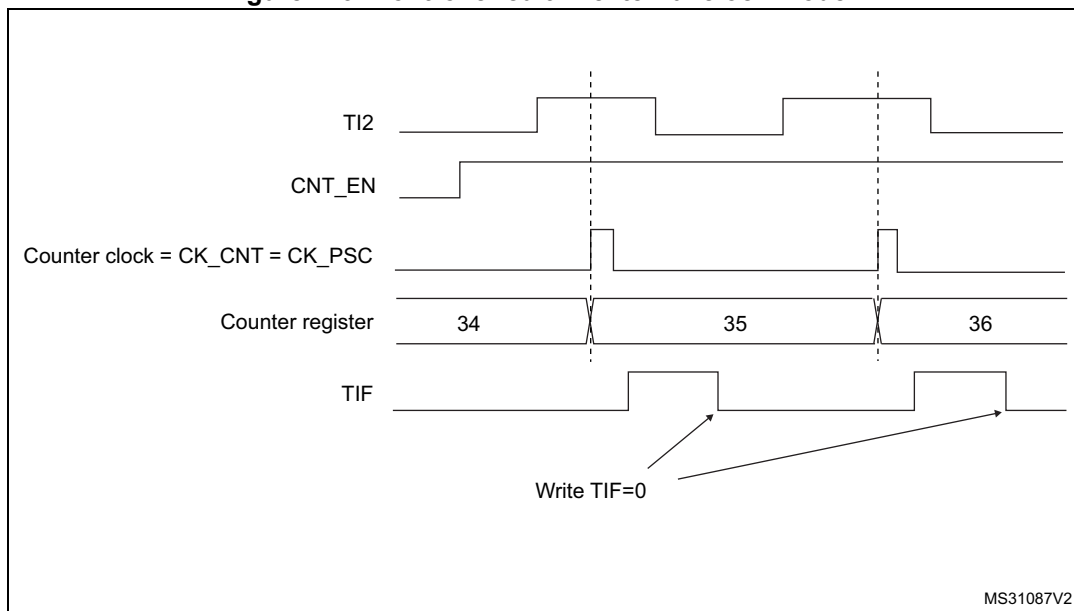
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx\_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the 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 249. Control circuit in external clock mode 1**



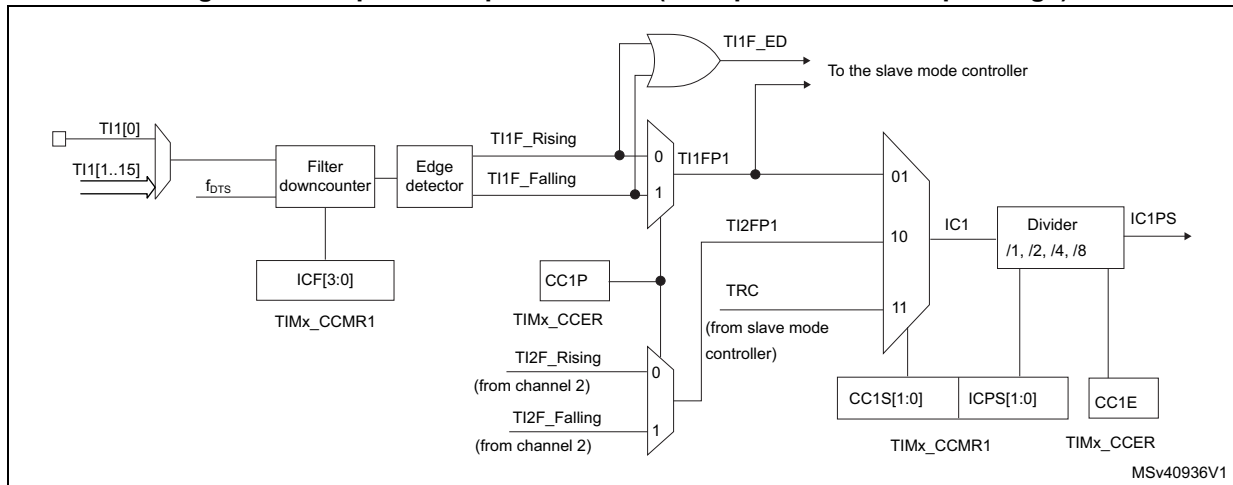
### 25.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

[Figure 250](#) to [Figure 253](#) 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 250. 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 251. Capture/compare channel 1 main circuit

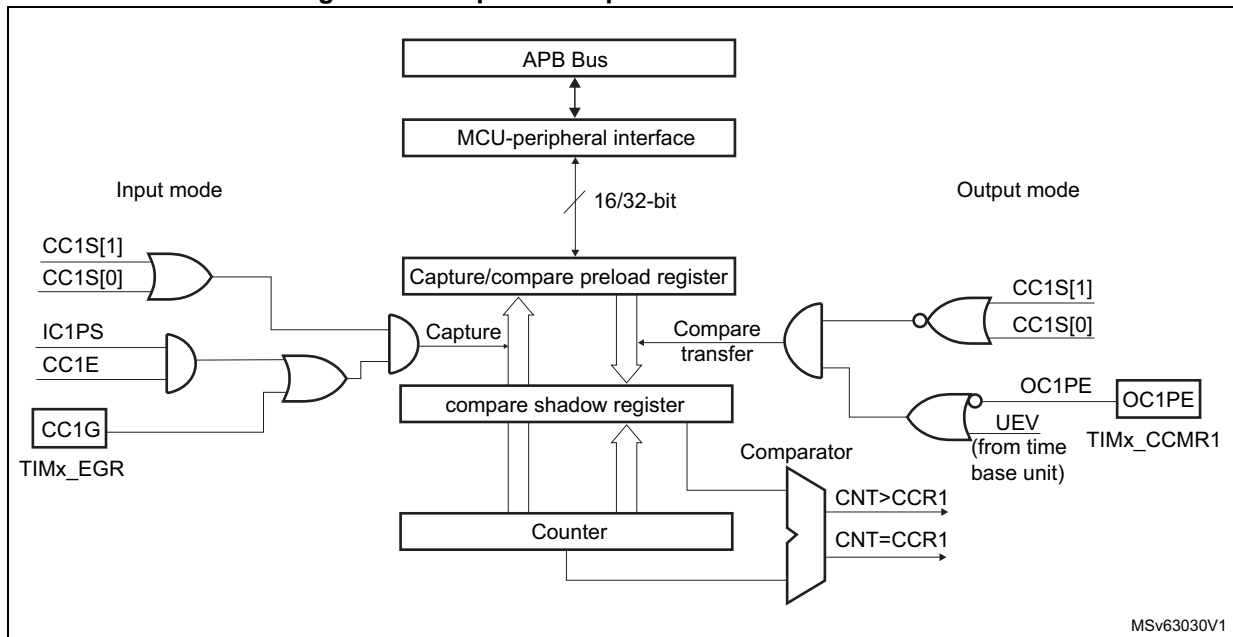


Figure 252. Output stage of capture/compare channel (channel 1)

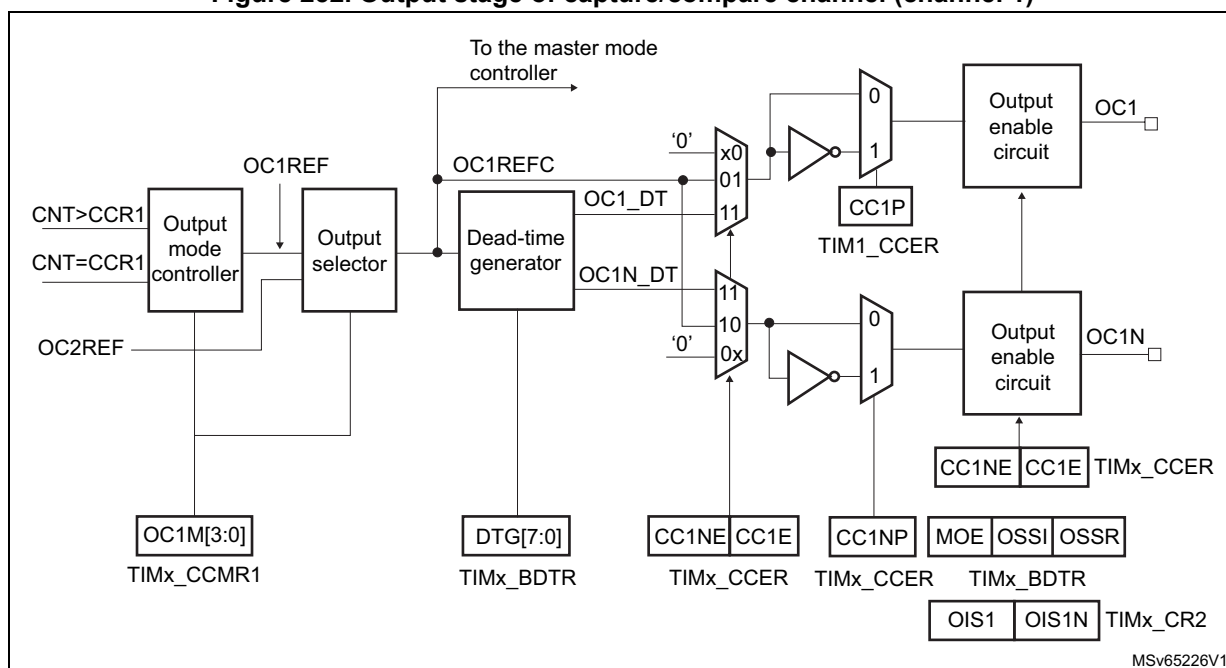
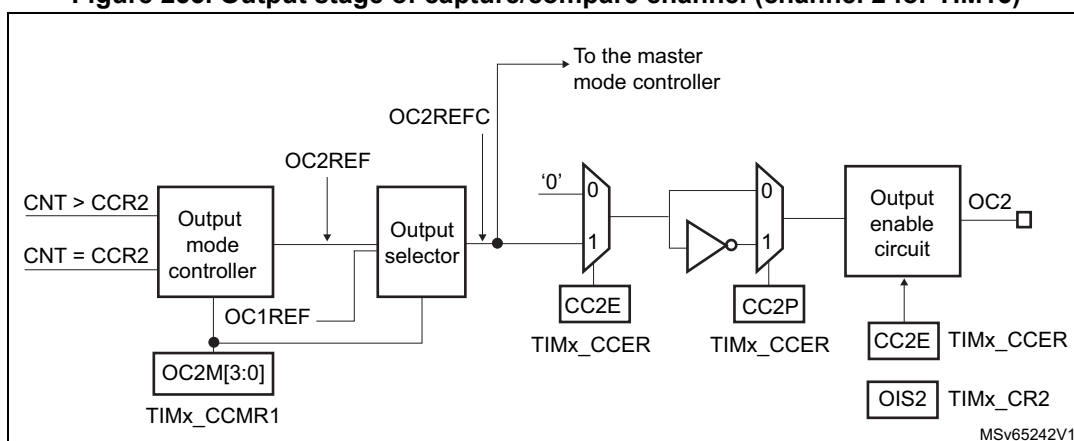


Figure 253. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 25.4.6 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx\_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCxIF flag (TIMx\_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was



already high, then the over-capture flag CCxOF (TIMx\_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx\_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx\_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input: TIMx\_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx\_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx\_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx\_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at  $f_{DTS}$  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.

#### 25.4.7 PWM input mode (only for TIM15)

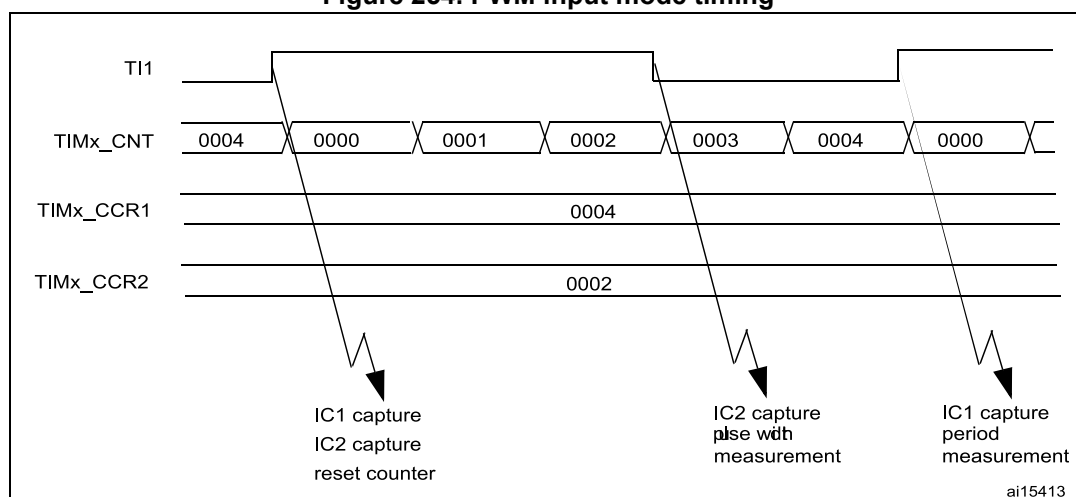
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx\_CCR1 register) and the duty cycle (in TIMx\_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK\_INT frequency and prescaler value):

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx\_TISEL register.
2. Select the active input for TIMx\_CCR1: write the CC1S bits to 01 in the TIMx\_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx\_CCR1 and counter clear): write the CC1P and CC1NP bits to '0' (active on rising edge).
4. Select the active input for TIMx\_CCR2: write the CC2S bits to 10 in the TIMx\_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx\_CCR2): write the CC2P and CC2NP bits to '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 100 in the TIMx\_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to '1' in the TIMx\_CCER register.

**Figure 254. 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.

### 25.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx\_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx\_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx\_CCMRx register.

Anyway, the comparison between the TIMx\_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 25.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx\_CCMRx register) and the output polarity (CCxP bit in the TIMx\_CCER register). The output pin can keep its level (OCxM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx\_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx\_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx\_DIER register, CCDS bit in the TIMx\_CR2 register for the DMA request selection).

The TIMx\_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx\_CCMRx register.

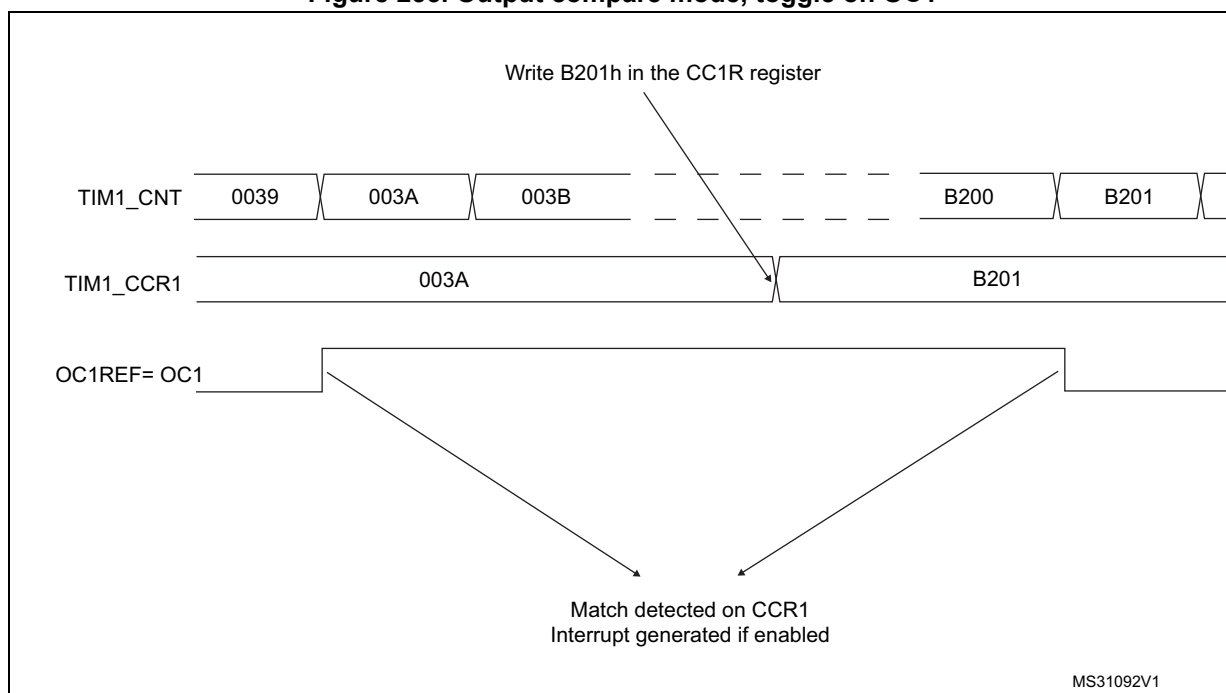
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

#### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx\_ARR and TIMx\_CCRx registers.
3. Set the CCXIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
  - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
  - Write OCxPE = 0 to disable preload register
  - Write CCxP = 0 to select active high polarity
  - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx\_CR1 register.

The TIMx\_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx\_CCRx shadow register is updated only at the next update event UEV). An example is given in [Figure 255](#).

Figure 255. Output compare mode, toggle on OC1



### 25.4.10 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx\_ARR register and a duty cycle determined by the value of the TIMx\_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx\_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx\_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx\_CCER and TIMx\_BDTR registers). Refer to the TIMx\_CCER register description for more details.

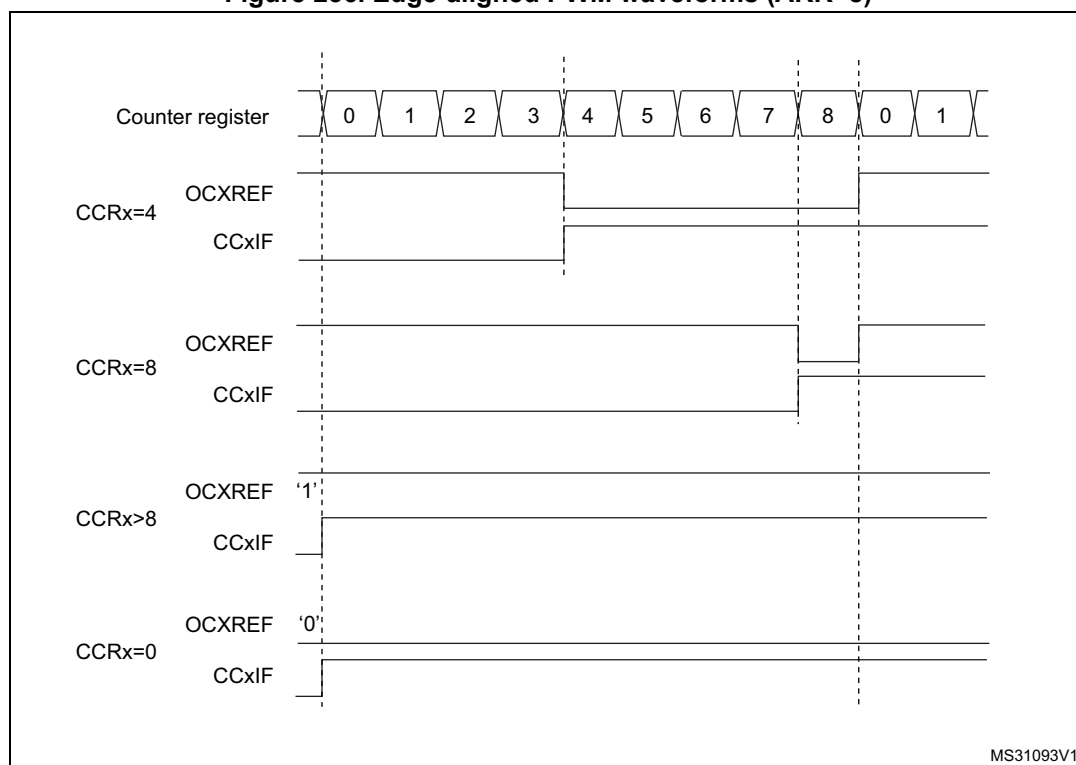
In PWM mode (1 or 2), TIMx\_CNT and TIMx\_CCRx are always compared to determine whether  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to [Upcounting mode on page 739](#).

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 256](#) shows some edge-aligned PWM waveforms in an example where TIMx\_ARR=8.

**Figure 256. Edge-aligned PWM waveforms (ARR=8)**



### 25.4.11 Combined PWM mode (TIM15 only)

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 the TIMx\_CCR1 and TIMx\_CCR2 registers

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.

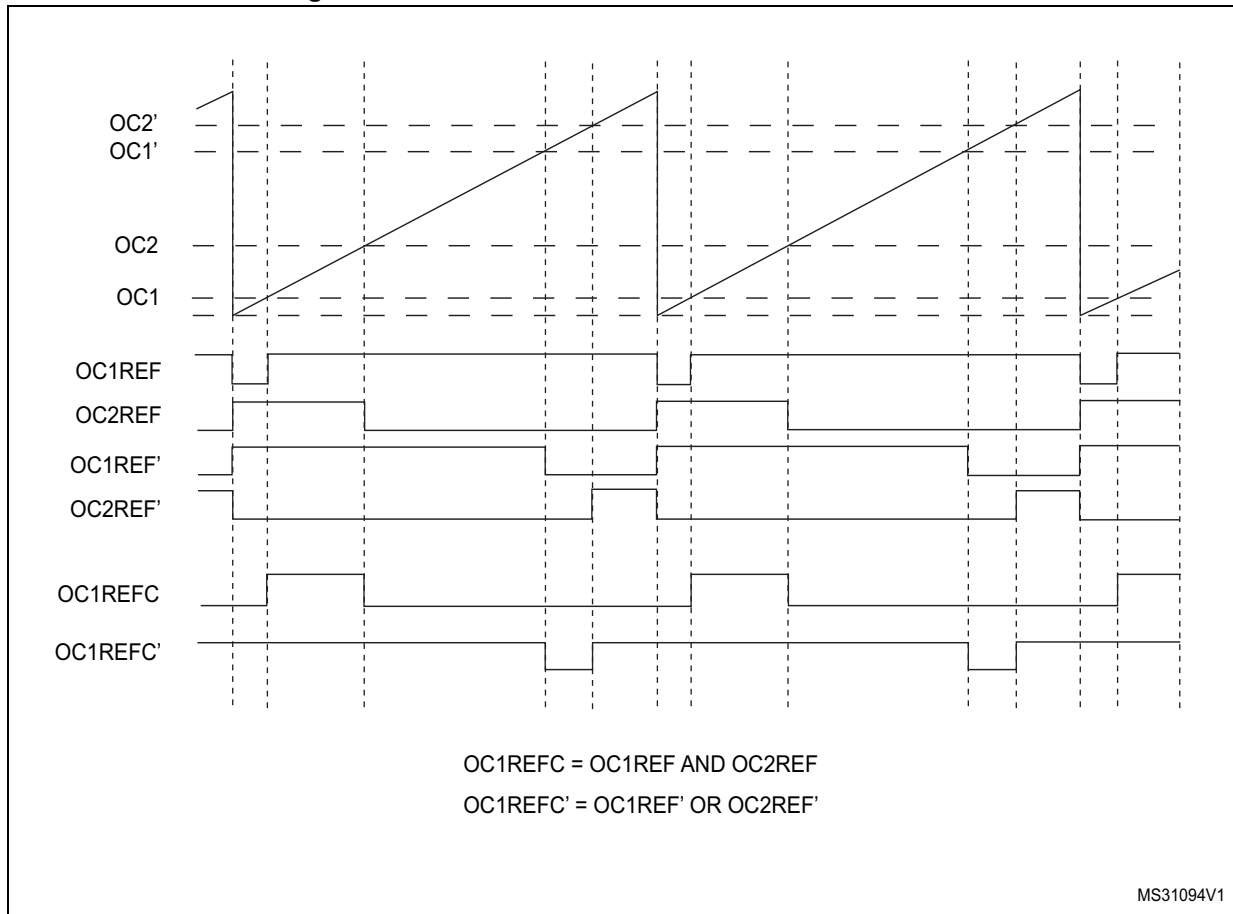
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*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 257](#) 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,

Figure 257. Combined PWM mode on channel 1 and 2



### 25.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx\_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx\_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx\_BDTR and TIMx\_CR2 registers. Refer to [Table 133: Output control bits for complementary OCx and OCxN channels with break feature \(TIM16/17\) on page 809](#) 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

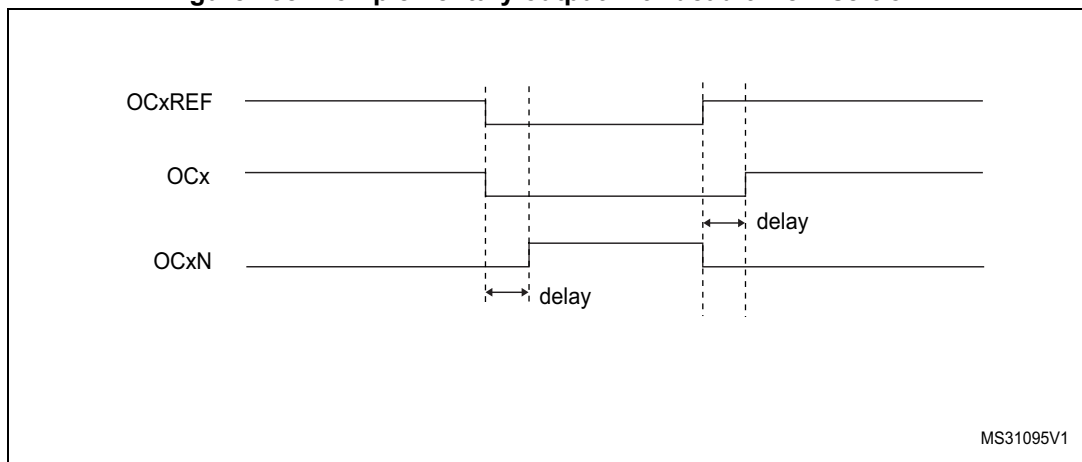
reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- 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.

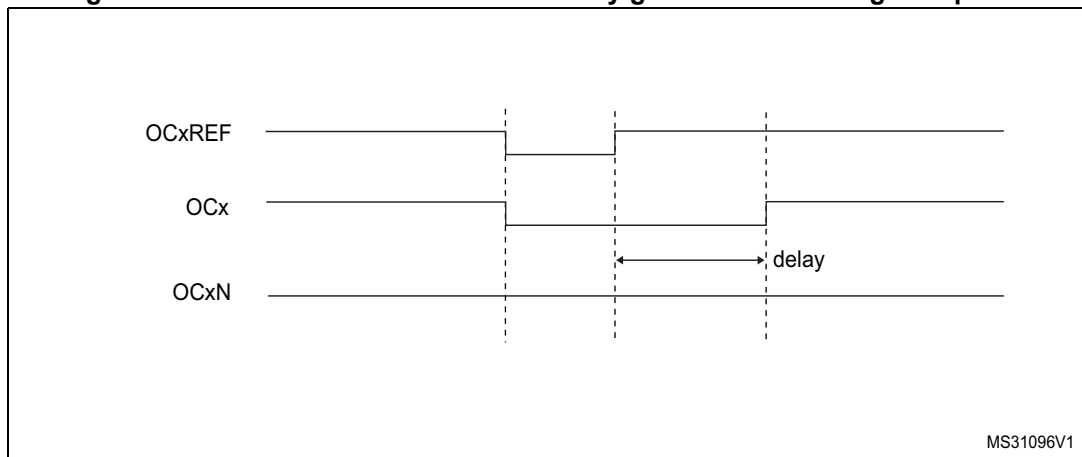
If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

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)

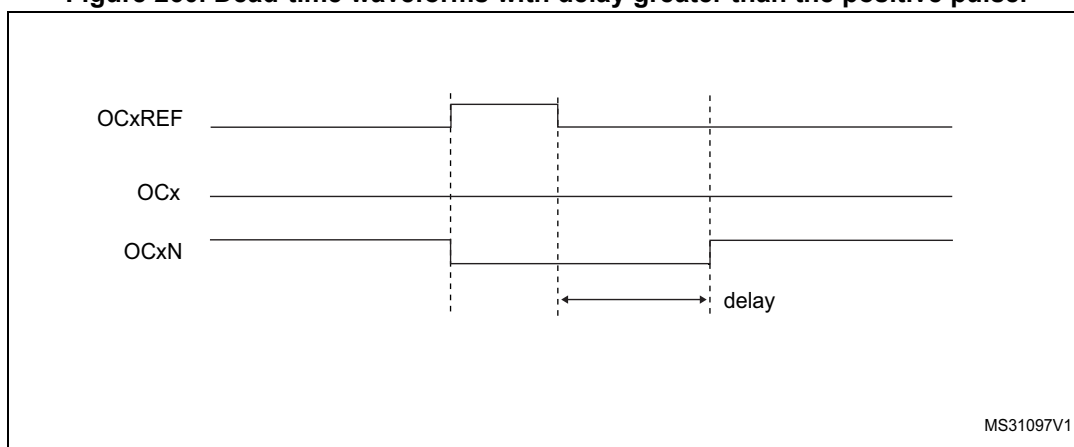
**Figure 258. Complementary output with dead-time insertion.**



**Figure 259. Dead-time waveforms with delay greater than the negative pulse.**



**Figure 260. 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 25.6.14: TIMx break and dead-time register \(TIMx\\_BDTR\)\(x = 16 to 17\) on page 812](#) 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.

### 25.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.



The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx\_BDTR register allows to enable /disable the outputs by software and is reset in case of break event.
- the OSSI bit in the TIMx\_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx\_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to [Table 131: Output control bits for complementary OCx and OCxN channels with break feature \(TIM15\) on page 787](#) 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.

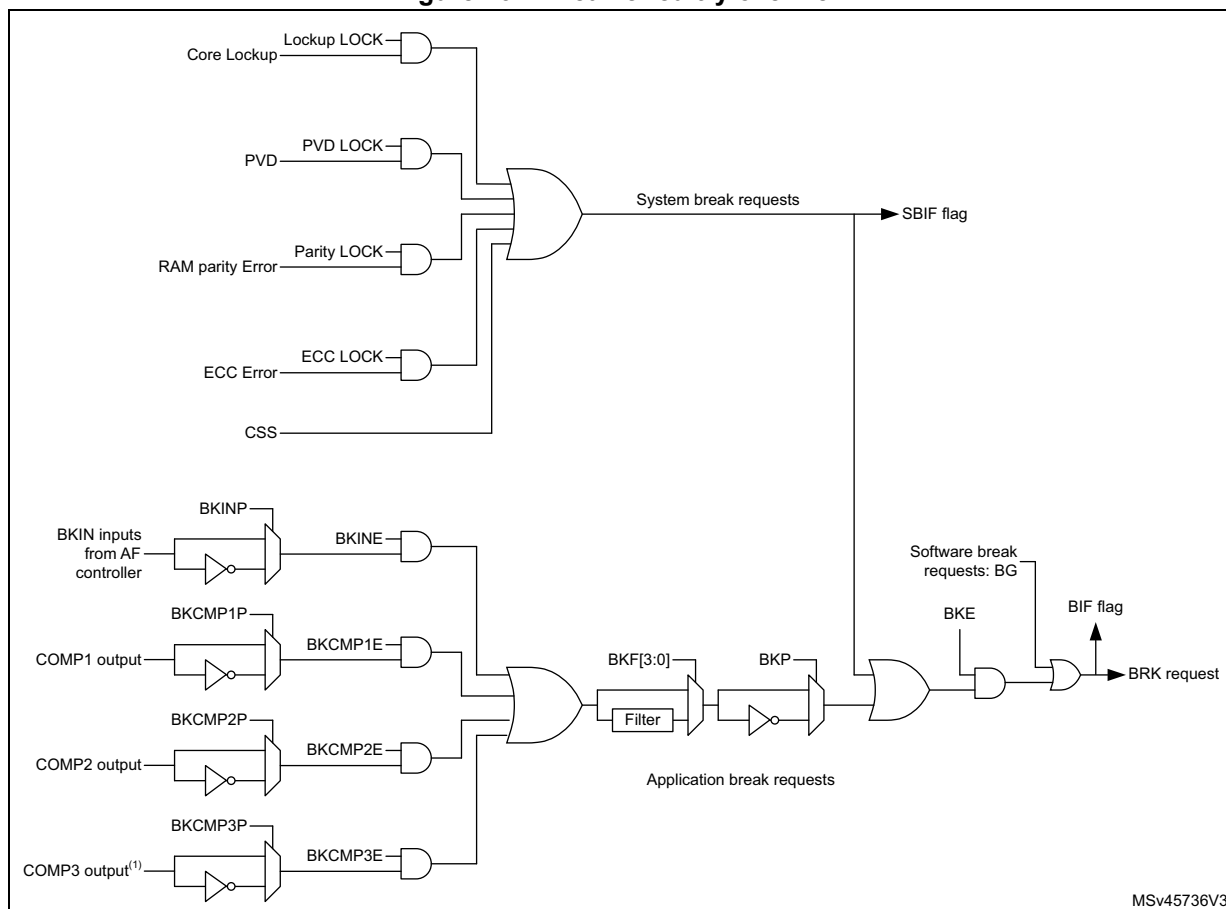
A programmable filter (BKF[3:0] bits in the TIMx\_BDTR register allows to filter out spurious events.

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:
  - the output from a comparator, with polarity selection and optional digital filtering
  - A system break:
    - the Cortex<sup>®</sup>-M0+ LOCKUP output
    - the PVD output
    - the SRAM parity error signal
    - a flash ECC error
    - a clock failure event generated by the CSS detector

Figure 261. Break circuitry overview



1. Available on STM32G0B1xx and STM32G0C1xx salestypes only.

**Caution:** An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the 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 OSSI=0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx\_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx\_DIER register is set.
- If the AOE bit in the TIMx\_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

*Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value.  
If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx\_CR2 register.*

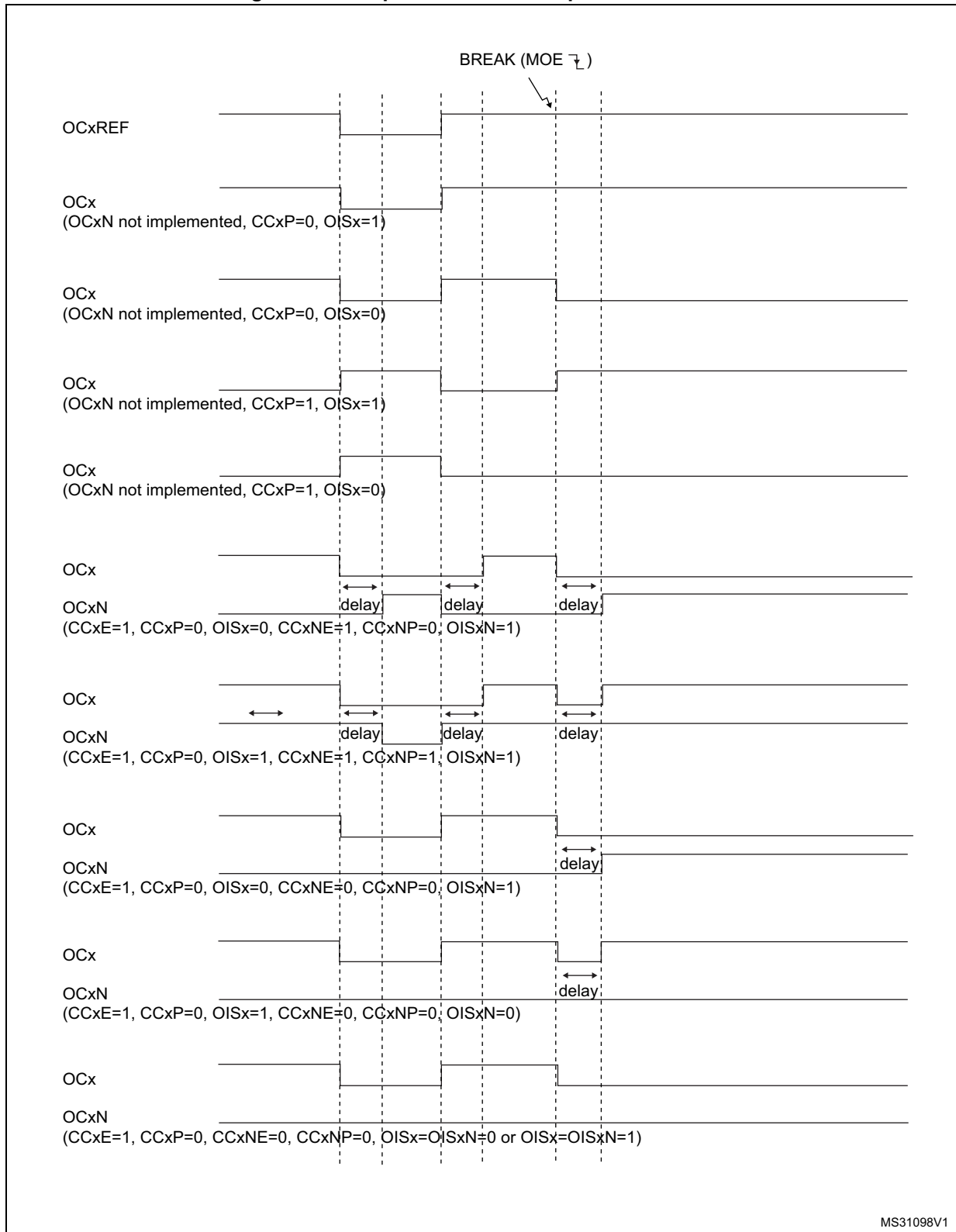
*Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.*

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx\_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be frozen (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx\_BDTR register. Refer to [Section 25.6.14: TIMx break and dead-time register \(TIMx\\_BDTR\)\(x = 16 to 17\) on page 812](#). The LOCK bits can be written only once after an MCU reset.

The [Figure 262](#) shows an example of behavior of the outputs in response to a break.

Figure 262. Output behavior in response to a break



### 25.4.14 Bidirectional break inputs

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on [Figure 263](#).

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 re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see [Table 129](#))

**Table 129. 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

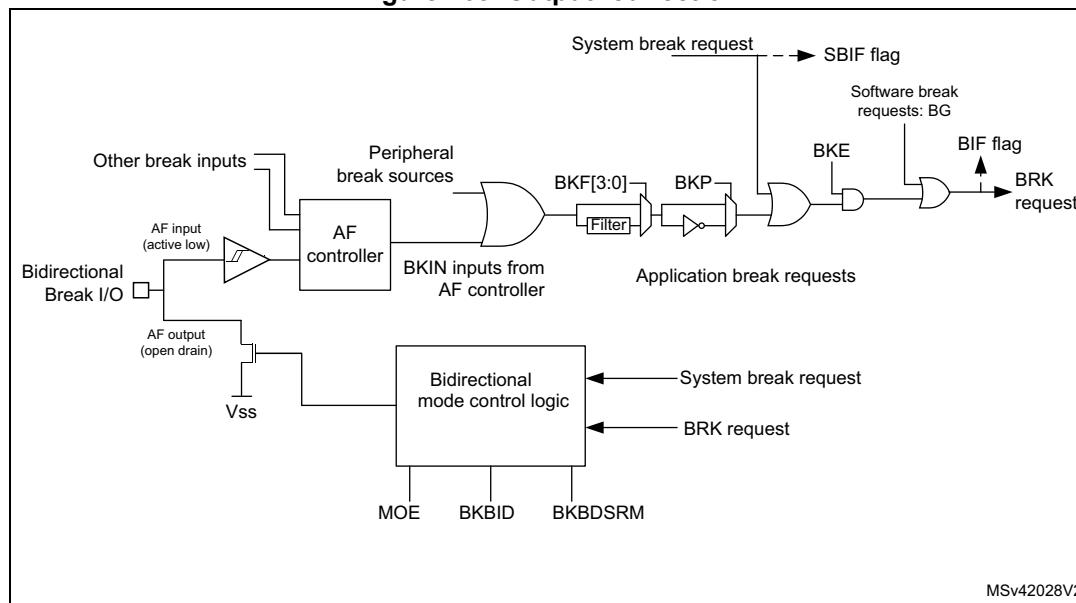
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 263. Output redirection



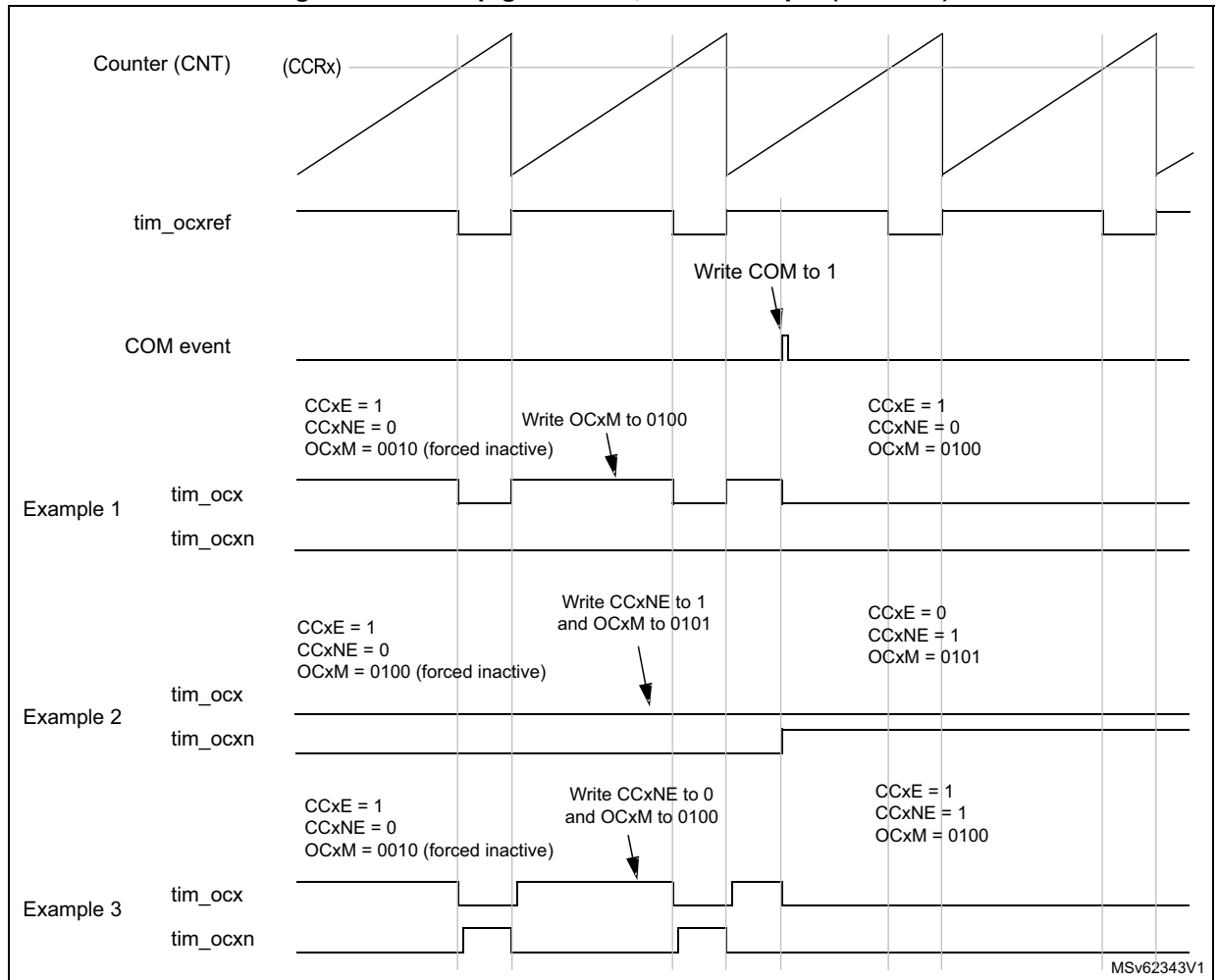
### 25.4.15 6-step PWM generation

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 264](#) describes the behavior of the tim\_ocx and tim\_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 264. 6-step generation, COM example (OSSR=1)



### 25.4.16 One-pulse mode

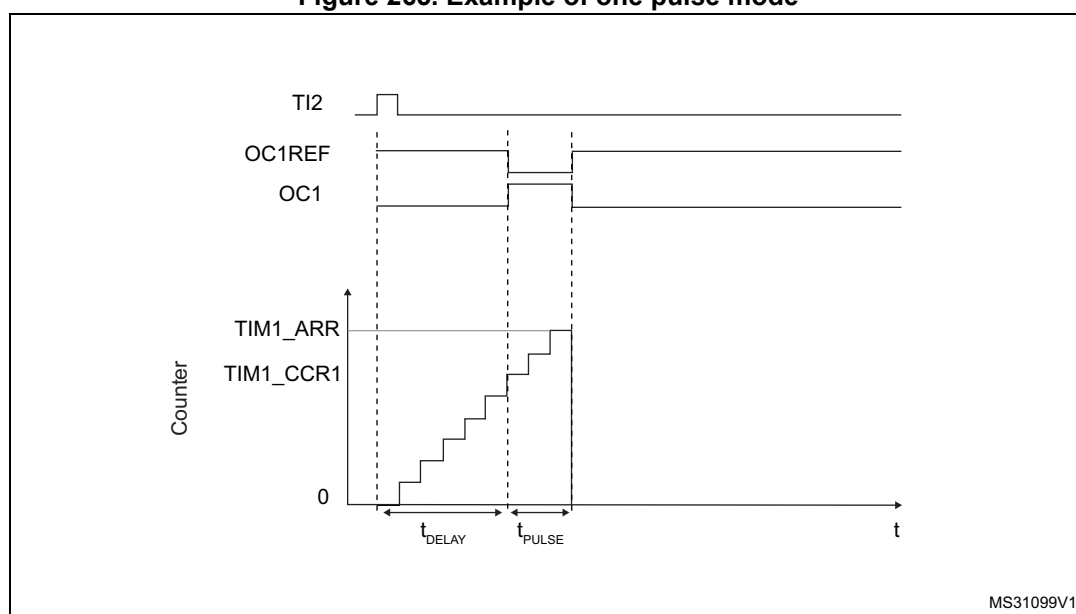
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx\_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $CNT < CCRx \leq ARR$  (in particular,  $0 < CCRx$ )

**Figure 265. Example of one pulse mode**



For example one may want to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  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  $t_{\text{DELAY}}$  is defined by the value written in the TIMx\_CCR1 register.
- The  $t_{\text{PULSE}}$  is defined by the difference between the auto-reload value and the compare value (TIMx\_ARR - TIMx\_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx\_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx\_CCMR1 register and ARPE in the TIMx\_CR1 register. In this case one has to write the compare value in the TIMx\_CCR1 register, the auto-reload value in the TIMx\_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx\_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay  $t_{\text{DELAY min}}$  we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx\_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 25.4.17 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in [Section 25.4.16](#):

- 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

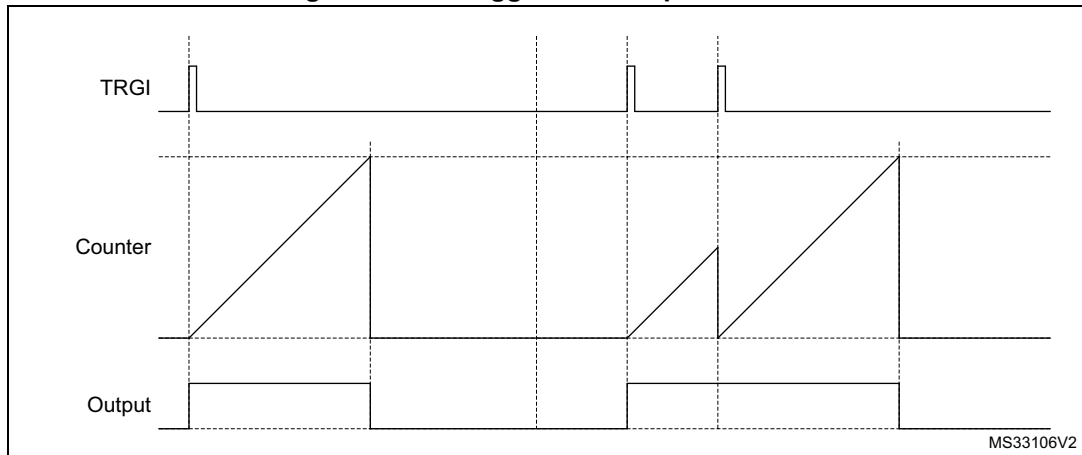
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx\_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

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 266. Retriggerable one pulse mode



**25.4.18 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).

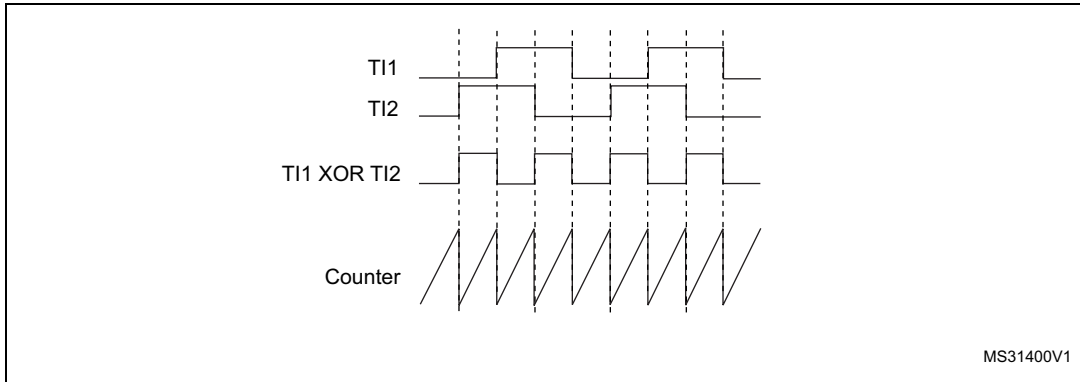
There is no latency between the assertions of the UIF and UIFCPY flags.

**25.4.19 Timer input XOR function (TIM15 only)**

The TI1S bit in the TIMx\_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx\_CH1 and TIMx\_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in [Figure 267](#).

**Figure 267. Measuring time interval between edges on 2 signals**



### 25.4.20 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

#### Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx\_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx\_ARR, TIMx\_CCRx) are updated.

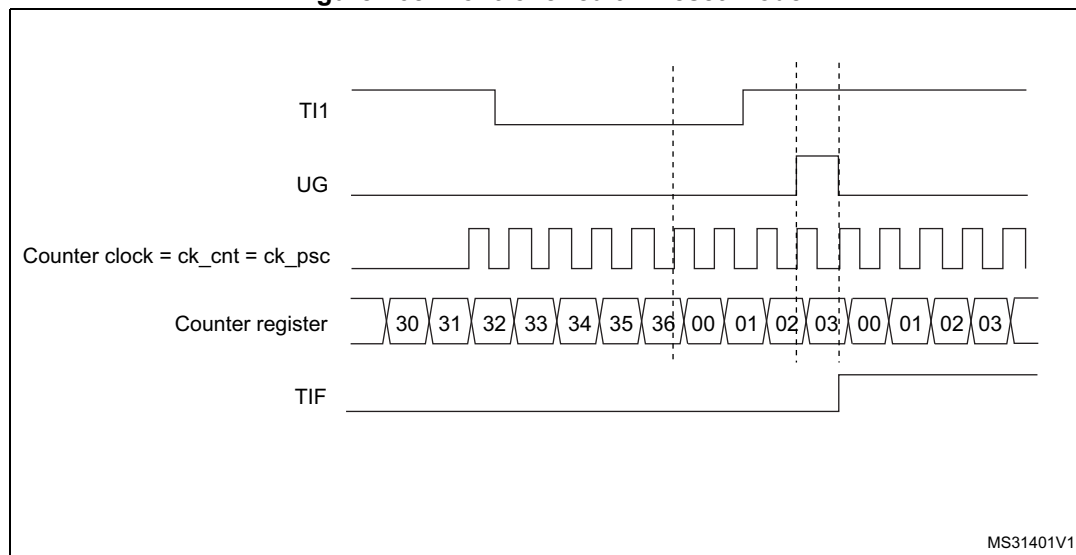
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx\_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx\_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx\_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx\_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx\_DIER register).

The following figure shows this behavior when the auto-reload register TIMx\_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 268. Control circuit in reset mode



**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

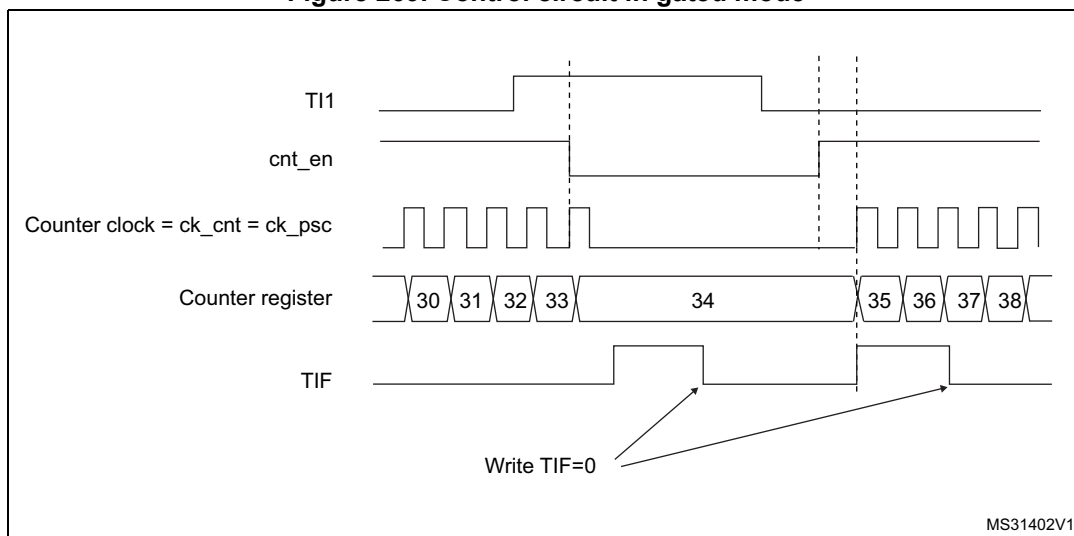
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx\_CCMR1 register. Write CC1P=1 and CC1NP = '0' in the TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx\_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx\_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx\_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx\_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

**Figure 269. Control circuit in gated mode**



### Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

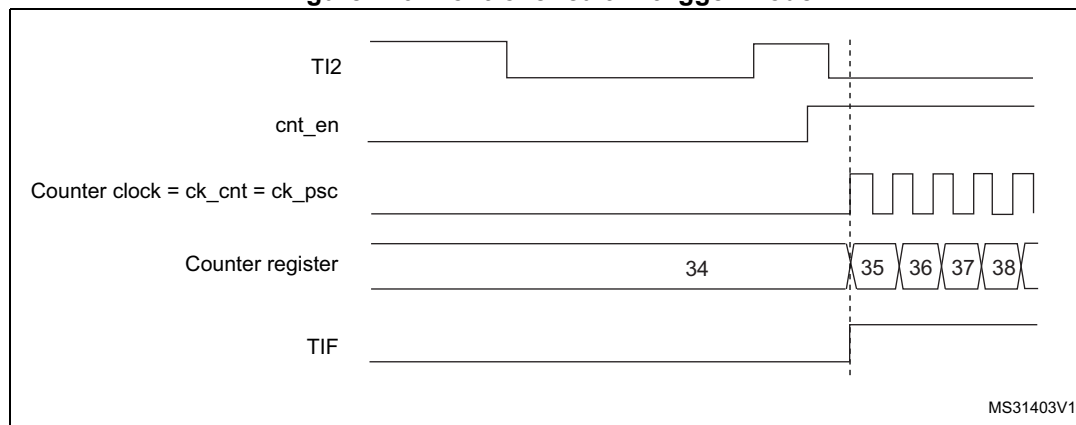
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx\_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx\_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx\_SMCR register. Select TI2 as the input source by writing TS=00110 in the TIMx\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

**Figure 270. Control circuit in trigger mode**



#### 25.4.21 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.

#### 25.4.22 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.

### 25.4.23 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to [Section 22.3.19: Timer synchronization](#) for details.

*Note:* The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### 25.4.24 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.

### 25.4.25 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 40.9.2: Debug support for timers, watchdog and I<sup>2</sup>C](#).

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.



## 25.5 TIM15 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).

### 25.5.1 TIM15 control register 1 (TIM15\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				r/w		r/w	r/w	r/w				r/w	r/w	r/w	r/w

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

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx\_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx\_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (CK\_INT) frequency and the dead-time and sampling clock ( $t_{DTS}$ ) used by the dead-time generators and the digital filters (Tix)

00:  $t_{DTS} = t_{CK\_INT}$

01:  $t_{DTS} = 2 * t_{CK\_INT}$

10:  $t_{DTS} = 4 * t_{CK\_INT}$

11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx\_ARR register is not buffered

1: TIMx\_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 25.5.2 TIM15 control register 2 (TIM15\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bit 10 **OIS2**: Output idle state 2 (OC2 output)

0: OC2=0 when MOE=0

1: OC2=1 when MOE=0

*Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15\_BDTR register).*

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0

1: OC1N=1 after a dead-time when MOE=0

*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15\_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 TIM15\_BDTR register).*

Bit 7 **TI1S**: TI1 selection

- 0: The TIMx\_CH1 pin is connected to TI1 input
- 1: The TIMx\_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

- 000: **Reset** - the UG bit from the TIMx\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
- 001: **Enable** - the Counter Enable signal CNT\_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is 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).

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.*

### 25.5.3 TIM15 slave mode control register (TIM15\_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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								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.

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

Bit 7 **MSM**: Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 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)

Other: Reserved

See [Table 130: TIMx Internal trigger connection on page 777](#) for more details on ITRx meaning for each Timer.

*Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.*

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (refer to ETP bit in TIMx\_SMCR for tim\_etr\_in and CCxP/CCxNP bits in TIMx\_CCER register for tim\_ti1fp1 and tim\_ti2fp2).

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

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.

Other codes: 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 130. TIMx Internal trigger connection**

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM15	TIM2	TIM3	TIM16_OC1	TIM17_OC1

### 25.5.4 TIM15 DMA/interrupt enable register (TIM15\_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw				rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled



Bits 12:10 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 **TIE**: Trigger interrupt enable  
 0: Trigger interrupt disabled  
 1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable  
 0: COM interrupt disabled  
 1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable  
 0: CC2 interrupt disabled  
 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable  
 0: CC1 interrupt disabled  
 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable  
 0: Update interrupt disabled  
 1: Update interrupt enabled

### 25.5.5 TIM15 status register (TIM15\_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

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

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag  
 Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag  
 This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.  
 0: No overcapture has been detected  
 1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

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.
- When CNT is reinitialized by a trigger event (refer to [Section 25.5.3: TIM15 slave mode control register \(TIM15\\_SMCR\)](#)), if URS=0 and UDIS=0 in the TIMx\_CR1 register.

### 25.5.6 TIM15 event generation register (TIM15\_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx\_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

*Note: This bit acts only on channels that have a complementary output.*

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

**If channel CC1 is configured as output:**

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

**If channel CC1 is configured as input:**

The current value of the counter is captured in TIMx\_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).



### 25.5.7 TIM15 capture/compare mode register 1 (TIM15\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

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]	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note:* CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{DTS}$   
 0001:  $f_{SAMPLING}=f_{CK\_INT}$ , N=2  
 0010:  $f_{SAMPLING}=f_{CK\_INT}$ , N=4  
 0011:  $f_{SAMPLING}=f_{CK\_INT}$ , N=8  
 0100:  $f_{SAMPLING}=f_{DTS}/2$ , N=6  
 0101:  $f_{SAMPLING}=f_{DTS}/2$ , N=8  
 0110:  $f_{SAMPLING}=f_{DTS}/4$ , N=6  
 0111:  $f_{SAMPLING}=f_{DTS}/4$ , N=8  
 1000:  $f_{SAMPLING}=f_{DTS}/8$ , N=6  
 1001:  $f_{SAMPLING}=f_{DTS}/8$ , N=8  
 1010:  $f_{SAMPLING}=f_{DTS}/16$ , N=5  
 1011:  $f_{SAMPLING}=f_{DTS}/16$ , N=6  
 1100:  $f_{SAMPLING}=f_{DTS}/16$ , N=8  
 1101:  $f_{SAMPLING}=f_{DTS}/32$ , N=5  
 1110:  $f_{SAMPLING}=f_{DTS}/32$ , N=6  
 1111:  $f_{SAMPLING}=f_{DTS}/32$ , N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input  
 01: capture is done once every 2 events  
 10: capture is done once every 4 events  
 11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output  
 01: CC1 channel is configured as input, IC1 is mapped on TI1  
 10: CC1 channel is configured as input, IC1 is mapped on TI2  
 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)

*Note:* CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).

### 25.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15\_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

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
Res.	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
	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 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, IC2 is mapped on TI2.

10: CC2 channel is configured as input, IC2 is mapped on TI1.

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx\_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx\_CCER).*

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

0011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx\_CNT<TIMx\_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx\_CNT<TIMx\_CCR1 else active.

1000: 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: Reserved,

1111: 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, 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.*

*On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.*

*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. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection  
 This bit-field defines the direction of the channel (input/output) as well as the used input.  
 00: CC1 channel is configured as output.  
 01: CC1 channel is configured as input, IC1 is mapped on TI1.  
 10: CC1 channel is configured as input, IC1 is mapped on TI2.  
 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx\_SMCR register)  
*Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx\_CCER).*

### 25.5.9 TIM15 capture/compare enable register (TIM15\_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.  
 Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity  
 Refer to CC1NP description  
 Bit 6 Reserved, must be kept at reset value.  
 Bit 5 **CC2P**: Capture/Compare 2 output polarity  
 Refer to CC1P description  
 Bit 4 **CC2E**: Capture/Compare 2 output enable  
 Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high

1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

*Note: 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 TlxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TlxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TlxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TlxFP1 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 131](#) for details.

**Table 131. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)**

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)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCx and OCxN both in active state	
			1	0		
			1	1		
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.*

### 25.5.10 TIM15 counter (TIM15\_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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

### 25.5.11 TIM15 prescaler (TIM15\_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 25.5.12 TIM15 auto-reload register (TIM15\_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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 25.4.1: Time-base unit on page 737](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.



### 25.5.13 TIM15 repetition counter register (TIM15\_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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.

### 25.5.14 TIM15 capture/compare register 1 (TIM15\_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **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).

### 25.5.15 TIM15 capture/compare register 2 (TIM15\_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

**If channel CC2 is configured as output:**

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx\_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx\_CNT and signalled on OC2 output.

**If channel CC2 is configured as input:**

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

### 25.5.16 TIM15 break and dead-time register (TIM15\_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.	Res.	BKBID	Res.	BK DSRM	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, OSSI, 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 the BRK input signal and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously

0001:  $f_{\text{SAMPLING}} = f_{\text{CK\_INT}}$ , 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 the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register)

See OC/OCN enable description for more details ([Section 25.5.9: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 785](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 13 **BKP**: Break polarity

- 0: Break input BRK is active low
- 1: Break input BRK is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

- 0: Break inputs (BRK and CCS clock failure event) disabled
- 1: Break inputs (BRK and CCS clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 25.5.9: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 785](#)).

- 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 25.5.9: TIM15 capture/compare enable register \(TIM15\\_CCER\) on page 785](#)).

- 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).*

### 25.5.17 TIM15 DMA control register (TIM15\_DCR)

Address offset: 0x48

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit 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).

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,

...

### 25.5.18 TIM15 DMA address for full transfer (TIM15\_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address  
(TIMx\_CR1 address) + (DBA + DMA index) x 4

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 25.5.19 TIM15 alternate register 1 (TIM15\_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.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BKCM P3P	BKCM P2P	BKCM P1P	BKINP	Res.	Res.	Res.	Res.	Res.	BKCM P3E	BKCM P2E	BKCM P1E	BKINE
			rW	rW	rW	rW						rW	rW	rW	rW

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BKCM P3P**: BRK COMP3 input polarity

This bit selects the COMP3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP3 input is active low

1: COMP3 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 11 **BKCM P2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low

1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BKCM P1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low

1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BKCOMP3E**: BRK COMP3 enable

This bit enables the COMP3 for the timer's BRK input. COMP3 output is 'ORed' with the other BRK sources.

0: COMP3 input disabled

1: COMP3 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 2 **BKCOMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCOMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 25.5.20 TIM15 input selection register (TIM15\_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.	Res.	Res.
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 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM15\_CH2 input

0001: TIM2\_IC2

0010: TIM3\_IC2

Others: Reserved

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



Bits 3:0 **T11SEL[3:0]**: selects T11[0] to T11[15] input  
 0000: TIM15\_CH1 input  
 0001: TIM2\_IC1  
 0010: TIM3\_IC1  
 Others: Reserved

### 25.5.21 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 132. TIM15 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	<b>TIM15_CR1</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]	ARPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																						0												
0x04	<b>TIM15_CR2</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OISIN	OIS1	T11S	MMS[2:0]			CCDS	CCUS	UDIS	CCPC		
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	<b>TIM15_SMCR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]	Res.	Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]				
	Reset value											0	0				0										0	0	0			0	0	0	
0x0C	<b>TIM15_DIER</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE	
	Reset value																		0	0				0	0	0	0	0				0	0	0	
0x10	<b>TIM15_SR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF	
	Reset value																						0	0		0	0	0				0	0	0	
0x14	<b>TIM15_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.	BG	TG	COMG	Res.	Res.	Res.	CC2G	CC1G	UG	
	Reset value																									0	0	0				0	0	0	
0x18	<b>TIM15_CCMR1</b> Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [2:0]	Res.	Res.	OC2PE	OC2FE	CC2S [1:0]	Res.	OC1M [2:0]	OC1PE	OC1FE	CC1S [1:0]	Res.	Res.	Res.	Res.	Res.	
	Reset value								0										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
	<b>TIM15_CCMR1</b> Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]	Res.	Res.	IC2 PSC [1:0]	CC2S [1:0]	Res.	IC1F[3:0]	IC1 PSC [1:0]	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0





Table 132. TIM15 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						
0x20	TIM15_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E						
	Reset value																									0	0	0	0	0	0	0	0						
0x24	TIM15_CNT	UIFCPY or Res.																CNT[15:0]																					
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x28	TIM15_PSC	Res.																PSC[15:0]																					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x2C	TIM15_ARR	Res.																ARR[15:0]																					
	Reset value																		1	1	1	1	1	1	1	1	1	1	1	1	1	1							
0x30	TIM15_RCR	Res.																REP[7:0]																					
	Reset value																									0	0	0	0	0	0	0							
0x34	TIM15_CCR1	Res.																CCR1[15:0]																					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x38	TIM15_CCR2	Res.																CCR2[15:0]																					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x44	TIM15_BDTR	Res.																BKBID			BKDSRM			BKF[3:0]			MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]	DTG[7:0]					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x48	TIM15_DCR	Res.																DBL[4:0]				Res.						DBA[4:0]											
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x4C	TIM15_DMAR	Res.																DMAB[15:0]																					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x60	TIM15_AF1	Res.																BKCMP3P			BKCMP2P			BKCMP1P			BKINP			Res.									
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0							
0x68	TIM15_TISEL	Res.																Res.						Res.						Res.									
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0							



Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

## 25.6 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).

### 25.6.1 TIMx control register 1 (TIMx\_CR1)(x = 16 to 17)

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	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.*

### 25.6.2 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
						rw	rw					rw	rw		rw

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.*

**25.6.3 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
						rw	rw	rw		rw				rw	rw

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

### 25.6.4 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
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, 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.

### 25.6.5 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.	Res.	Res.	Res.	Res.	Res.	Res.	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).

### 25.6.6 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.	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.	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 T11 input and the length of the digital filter applied to T11. 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).*

### 25.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx\_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

#### Output compare mode:

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.	OC1M [3]
															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.	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).*

**25.6.8 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

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

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

- 0: OC1N active high
- 1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register) and CC1S="00" (the channel is configured in output).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx\_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.*

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to [Table 133](#) for details.

**Table 133. 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		
	1		0	1	Off-State (output enabled with inactive state)	
			1	0	Asynchronously: OCx=CCxP, OCxN=CCxNP	
			1	1	Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	

1. When both outputs of a channel are not used (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.*

**25.6.9 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

### 25.6.10 TIMx prescaler (TIMx\_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK\_CNT) is equal to  $f_{CK\_PSC} / (PSC[15:0] + 1)$ .

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx\_EGR register or through trigger controller when configured in “reset mode”).

### 25.6.11 TIMx auto-reload register (TIMx\_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 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 25.4.1: Time-base unit on page 737](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

### 25.6.12 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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.

### 25.6.13 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]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 15:0 **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).

### 25.6.14 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.	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, OSSI, 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 OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx\_CCER register)

See OC/OCN enable description for more details ([Section 25.6.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 807](#)).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1: Break inputs (BRK and CCS clock failure event) enabled

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 25.6.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 807](#)).

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 25.6.8: TIMx capture/compare enable register \(TIMx\\_CCER\)\(x = 16 to 17\) on page 807](#)).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx\_BDTR register, OISx and OISxN bits in TIMx\_CR2 register and BKE/BKP/AOE bits in TIMx\_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx\_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx\_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx\_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x  $t_{dtg}$  with  $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x  $t_{dtg}$  with  $t_{dtg} = 2 \times t_{DTS}$

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x  $t_{dtg}$  with  $t_{dtg} = 8 \times t_{DTS}$

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x  $t_{dtg}$  with  $t_{dtg} = 16 \times t_{DTS}$

Example if  $t_{DTS} = 125$  ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16  $\mu$ s to 31750 ns by 250 ns steps,

32  $\mu$ s to 63  $\mu$ s by 1  $\mu$ s steps,

64  $\mu$ s to 126  $\mu$ s by 2  $\mu$ s steps

*Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 25.6.15 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.	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.

### 25.6.16 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  
 $(\text{TIMx\_CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4$

where TIMx\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx\_DCR).

### 25.6.17 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.	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.	BKCM P3P	BKCM P2P	BKCM P1P	BKINP	Res.	Res.	Res.	Res.	Res.	BKCM P3E	BKCM P2E	BKCM P1E	BKINE
			rw	rw	rw	rw						rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

**Bit 12 BKCM3P:** BRK COMP3 input polarity

This bit selects the COMP3 input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: COMP3 input is active low
- 1: COMP3 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).*

*Note: This bit is available on STM32G0B1xx and STM32G0C1xx saletypes only, reserved otherwise.*

**Bit 11 BKCM2P:** BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: COMP2 input is active low
- 1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 10 BKCM1P:** BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: COMP1 input is active low
- 1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

**Bit 9 BKINP:** BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: BKIN input is active low
- 1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BKCOMP3E**: BRK COMP3 enable

This bit enables the COMP3 for the timer's BRK input. COMP3 output is 'ORed' with the other BRK sources.

- 0: COMP3 input disabled
- 1: COMP3 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: This bit is available on STM32G0B1xx and STM32G0C1xx salestypes only, reserved otherwise.*

Bit 2 **BKCOMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

- 0: COMP2 input disabled
- 1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCOMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

- 0: COMP1 input disabled
- 1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

- 0: BKIN input disabled
- 1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 25.6.18 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.	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.	T11SEL[3:0]			
												rw	rw	rw	rw

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

Bits 3:0 **TI1SEL[3:0]**: selects T11[0] to T11[15] input  
 0000: TIM16\_CH1 input  
 0001: LSI  
 0010: LSE  
 0011: RTC wake-up  
 0100: MCO2<sup>(1)</sup>  
 Others: Reserved

1. Available on STM32G0B1xx and STM32G0C1xx salestypes only, reserved otherwise.

### 25.6.19 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.	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.	BKCM P3P	BKCM P2P	BKCM P1P	BKINP	Res.	Res.	Res.	Res.	Res.	BKCM P3E	BKCM P2E	BKCM P1E	BKINE
			rw	rw	rw	rw						rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BKCM P3P**: BRK COMP3 input polarity

This bit selects the COMP3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP3 input is active low  
 1: COMP3 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).*

*Note: This bit is available on STM32G0B1xx and STM32G0C1xx salestypes only, reserved otherwise.*

Bit 11 **BKCM P2P**: BRK COMP2 input polarity

This bit selects the COMP2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP2 input is active low  
 1: COMP2 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 10 **BKCM P1P**: BRK COMP1 input polarity

This bit selects the COMP1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: COMP1 input is active low  
 1: COMP1 input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BKCOMP3E**: BRK COMP3 enable

This bit enables the COMP3 for the timer's BRK input. COMP3 output is 'ORed' with the other BRK sources.

0: COMP3 input disabled

1: COMP3 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: This bit is available on STM32G0B1xx and STM32G0C1xx saletypes only, reserved otherwise.*

Bit 2 **BKCOMP2E**: BRK COMP2 enable

This bit enables the COMP2 for the timer's BRK input. COMP2 output is 'ORed' with the other BRK sources.

0: COMP2 input disabled

1: COMP2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 1 **BKCOMP1E**: BRK COMP1 enable

This bit enables the COMP1 for the timer's BRK input. COMP1 output is 'ORed' with the other BRK sources.

0: COMP1 input disabled

1: COMP1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).*

### 25.6.20 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.	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.	T17SEL[3:0]			
												rw	rw	rw	rw



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

Bits 3:0 **TI1SEL[3:0]**: selects T11[0] to T11[15] input

0000: TIM17\_CH1 input

0001: HSI48/256<sup>(1)</sup>

0010: HSE/32

0011: MCO

0100: MCO2<sup>(1)</sup>

Others: Reserved

1. Available on STM32G0B1xx and STM32G0C1xx salestypes only, reserved otherwise.



### 25.6.21 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

**Table 134. TIM16/TIM17 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UJFREMA	Res.	CK[1:0]D	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN	
	Reset value																					0		0	0	0			0	0	0	0	
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0	Res.	Res.	Res.	Res.	0	0	Res.	0
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCIDE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	Res.	
	Reset value																							0	0	0	Res.	0	Res.	Res.	Res.	Res.	Res.
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC10F	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	Res.	
	Reset value																							0	Res.	0	Res.	0	Res.	Res.	Res.	Res.	Res.
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	Res.	Res.	Res.	Res.	Res.
0x18	TIMx_CCMR1 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.	
	Reset value																	OC1M[3]										OC1M[2:0]					
	TIMx_CCMR1 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.	
Reset value																											IC1F[3:0]	IC1PSC[1:0]	CC1S[1:0]	CC1S[1:0]	CC1S[1:0]	CC1S[1:0]	
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x24	TIMx_CNT	UIFCOPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																															
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 134. 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.	REP[7:0]										
	Reset value																									0	0	0	0	0	0	0	0	0		
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x44	TIMx_BDTR	Res.	Res.	Res.	BKID		BKSRM									BKF[3:0]	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]	DTG[7:0]												
	Reset value				0		0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x48	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																			
0x4C	TIMx_DMAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x60	TIM16_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKMP3P	BKMP2P	BKMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0				
0x60	TIM17_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKMP3P	BKMP2P	BKMP1P	BKINP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0				
0x68	TIM16_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																			
0x68	TIM17_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 60](#) for the register boundary addresses.



## 26 Low-power timer (LPTIM)

### 26.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

### 26.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no embedded oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

## 26.3 LPTIM implementation

Table 135 describes LPTIM implementation on STM32G0x1 devices: the full set of features is implemented in LPTIM1. LPTIM2 supports a smaller set of features, but is otherwise identical to LPTIM1.

Table 135. STM32G0x1 LPTIM features

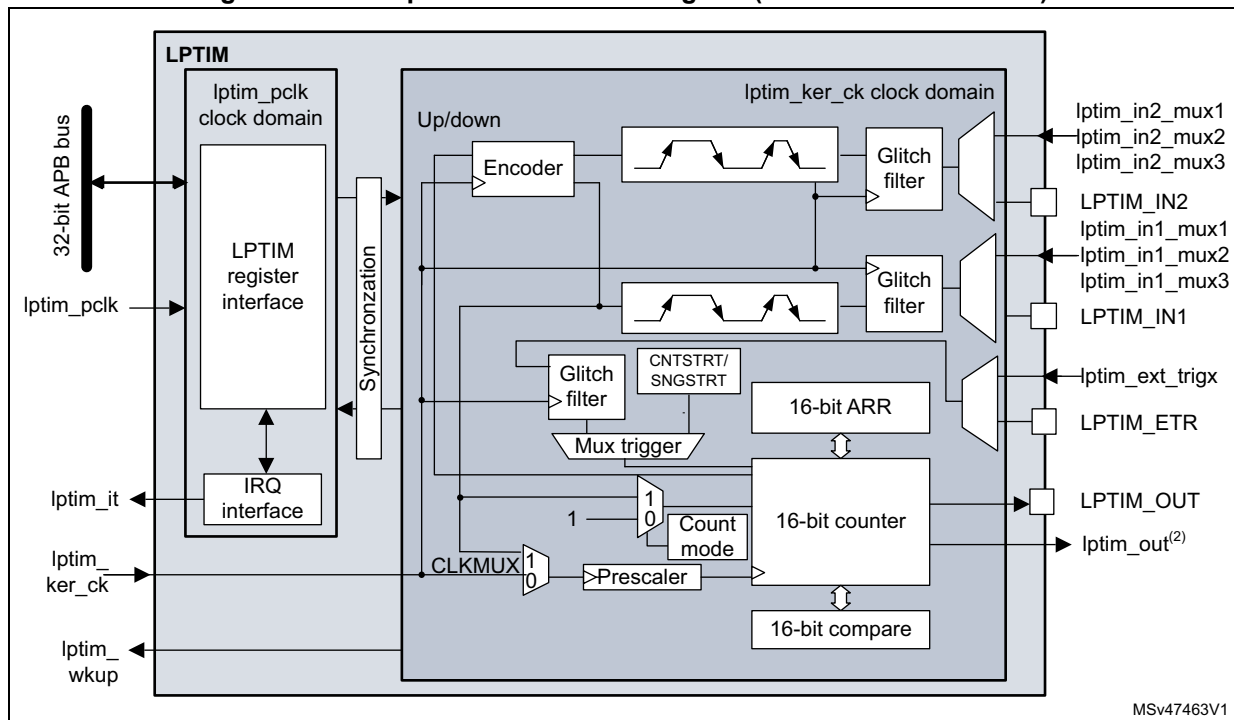
LPTIM modes/features <sup>(1)</sup>	LPTIM1	LPTIM2
Encoder mode	X	-

1. X = supported.

## 26.4 LPTIM functional description

### 26.4.1 LPTIM block diagram

Figure 271. Low-power timer block diagram (LPTIM1 and LPTIM2<sup>(1)</sup>)



1. LPTIM2 has only the input channel 1, no input channel 2
2. lptim\_out is the internal LPTIM output signal that can be connected to internal peripherals.

## 26.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

**Table 136. LPTIM input/output pins**

Names	Signal type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM Output GPIO pin

**Table 137. LPTIM internal signals**

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 <sup>(1)</sup> connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 <sup>(1)</sup> connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 <sup>(1)</sup> connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event

1. Only applies to LPTIM1

## 26.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

**Table 138. LPTIM1 external trigger connection**

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM1_ETR alternate function
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 input detection
lptim_ext_trig4	TAMP2 input detection
lptim_ext_trig5	COMP3_OUT

**Table 138. LPTIM1 external trigger connection (continued)**

TRIGSEL	External trigger
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

**Table 139. LPTIM2 external trigger connection**

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM2_ETR alternate function
lptim_ext_trig1	RTC ALARM A
lptim_ext_trig2	RTC ALARM B
lptim_ext_trig3	TAMP1 input detection
lptim_ext_trig4	TAMP2 input detection
lptim_ext_trig5	TAMP_TRG3
lptim_ext_trig6	COMP1_OUT
lptim_ext_trig7	COMP2_OUT

**Table 140. LPTIM1 input 1 connection**

lptim_in1_mux	LPTIM1 input 1 connected to
lptim_in1_mux0	GPIO pin as LPTIM1_IN1 alternate function
lptim_in1_mux1	COMP1_OUT
lptim_in1_mux2	Not connected
lptim_in1_mux3	Not connected

**Table 141. LPTIM1 input 2 connection**

lptim_in2_mux	LPTIM1 input 2 connected to
lptim_in2_mux0	GPIO pin as LPTIM1_IN2 alternate function
lptim_in2_mux1	COMP2_OUT
lptim_in2_mux2	Not connected
lptim_in2_mux3	Not connected

**Table 142. LPTIM2 input 1 connection**

lptim_in1_mux	LPTIM2 input 1 connected to
lptim_in1_mux0	GPIO pin as LPTIM2_IN1 alternate function
lptim_in1_mux1	COMP1_OUT
lptim_in1_mux2	COMP2_OUT
lptim_in1_mux3	COMP1_OUT OR COMP2_OUT

## 26.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM will use an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.

## 26.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

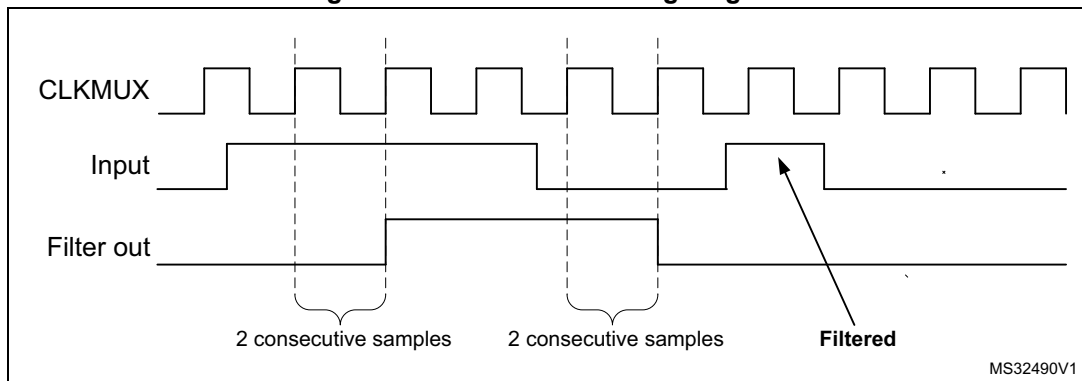
The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

*Note:* *The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.*

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. [Figure 272](#) shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 272. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

### 26.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

Table 143. Prescaler division ratios

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

### 26.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals '00', The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than '00', TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.



The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it will be ignored (unless timeout function is enabled).

*Note:* The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled will be discarded by hardware.

*Note:* When starting the counter by software ( $TRIGEN[1:0] = 00$ ), there is a delay of 3 kernel clock cycles between the LPTIM\_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

### 26.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when reaching the ARR value.

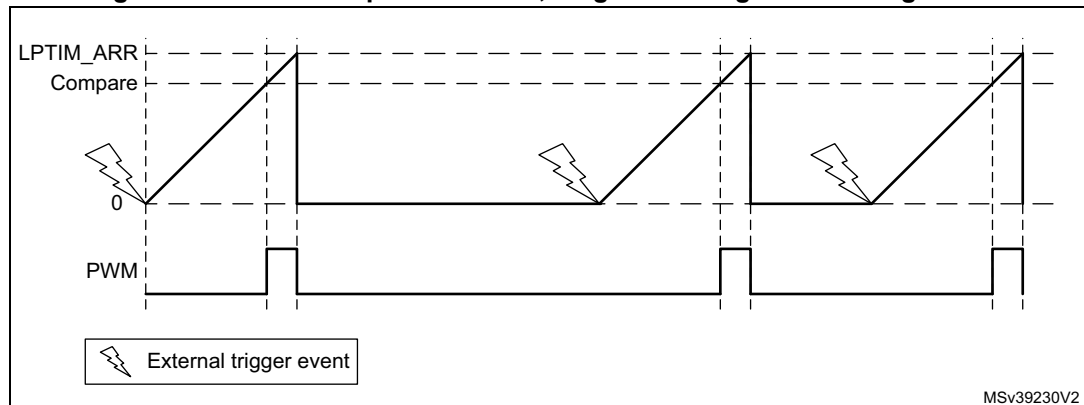
#### One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in [Figure 273](#).

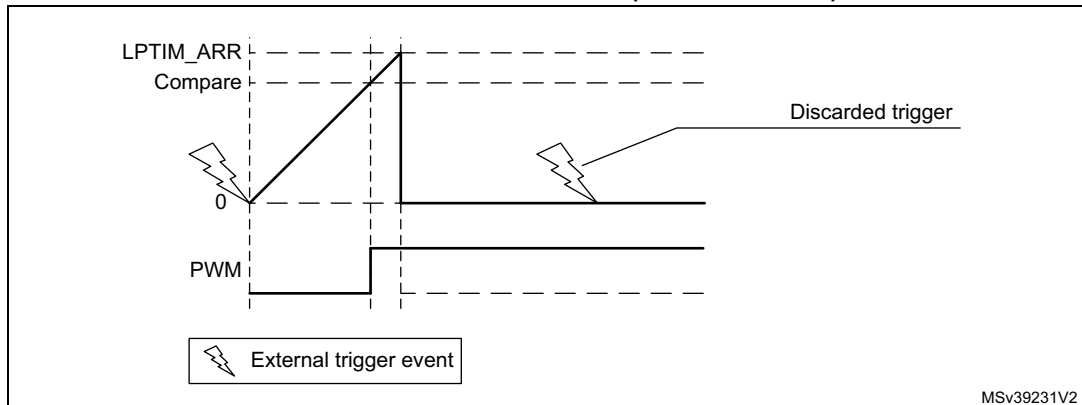
**Figure 273. LPTIM output waveform, single counting mode configuration**



Set-once mode activated:

It should be noted that when the WAVE bit-field in the LPTIM\_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in [Figure 274](#).

**Figure 274. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)**



In case of software start (TRIGEN[1:0] = '00'), the SNGSTRT setting will start the counter for one-shot counting.

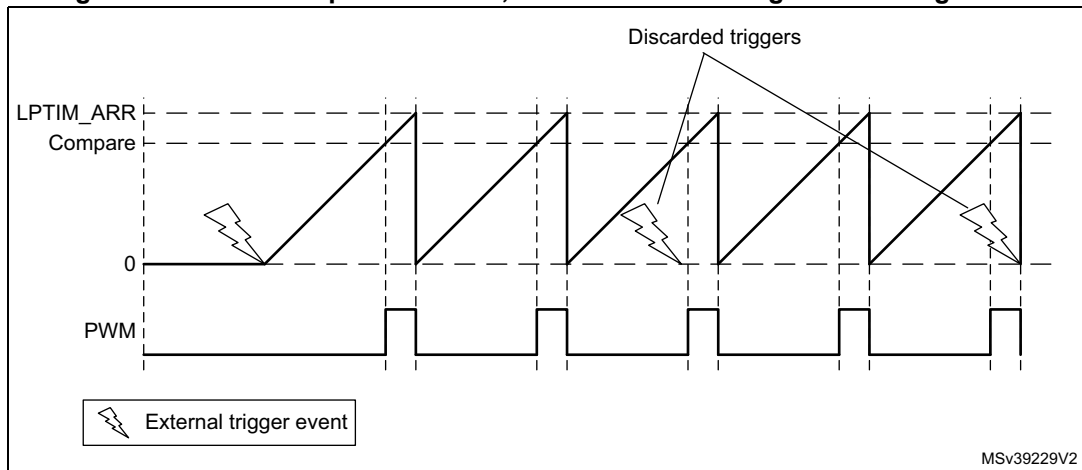
**Continuous mode**

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in [Figure 275](#).

In case of software start (TRIGEN[1:0] = '00'), setting CNTSTRT will start the counter for continuous counting.

**Figure 275. LPTIM output waveform, Continuous counting mode configuration**



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to '1'). It is possible to change "on the fly" from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.

### 26.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMEOUT bit.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

### 26.4.10 Waveform generation

Two 16-bit registers, the LPTIM\_ARR (autoreload register) and LPTIM\_CMP (compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM\_CNT exceeds the compare value in LPTIM\_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM\_ARR and the LPTIM\_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM\_ARR register value be strictly greater than the LPTIM\_CMP register value.

The LPTIM output waveform can be configured through the WAVE bit as follow:

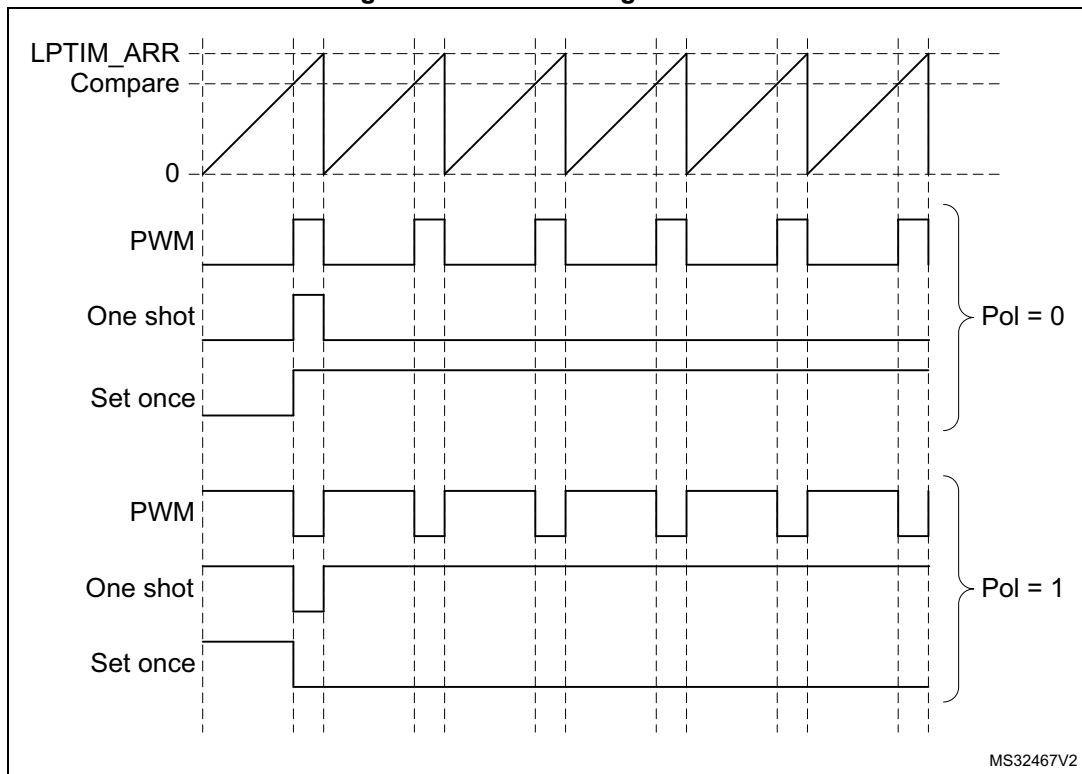
- Resetting the WAVE bit to '0' forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to '1' forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated.

[Figure 276](#) below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.

Figure 276. Waveform generation



### 26.4.11 Register update

The LPTIM\_ARR register and LPTIM\_CMP register are updated immediately after the APB bus write operation, or at the end of the current period if the timer is already started.

The PRELOAD bit controls how the LPTIM\_ARR and the LPTIM\_CMP registers are updated:

- When the PRELOAD bit is reset to '0', the LPTIM\_ARR and the LPTIM\_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to '1', the LPTIM\_ARR and the LPTIM\_CMP registers are updated at the end of the current period, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM\_ISR register indicate when the write operation is completed to respectively the LPTIM\_ARR register and the LPTIM\_CMP register.

After a write to the LPTIM\_ARR register or the LPTIM\_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, will lead to unpredictable results.

### 26.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source will be used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0  
The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1  
The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.  
Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source  
COUNTMODE value is don't care.  
In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.  
For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.  
Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

### 26.4.13 Timer enable

The ENABLE bit located in the LPTIM\_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM\_CFGR and LPTIM\_IER registers must be modified only when the LPTIM is disabled.

### 26.4.14 Timer counter reset

In order to reset the content of LPTIM\_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM\_CR register. After setting the COUNTRST bit-field to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic will elapse before the reset is taken into account. This will make the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.
- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM\_CR register. When this bit is set to '1', any read access to the LPTIM\_CNT register will reset its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle will occur on the LPTIM Input1.

It should be noted that to read reliably the content of the LPTIM\_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM\_CNT register.

---

**Warning:** There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.

---

### 26.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM\_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM\_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signaled by the two Down and Up flags in the LPTIM\_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to '1'. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

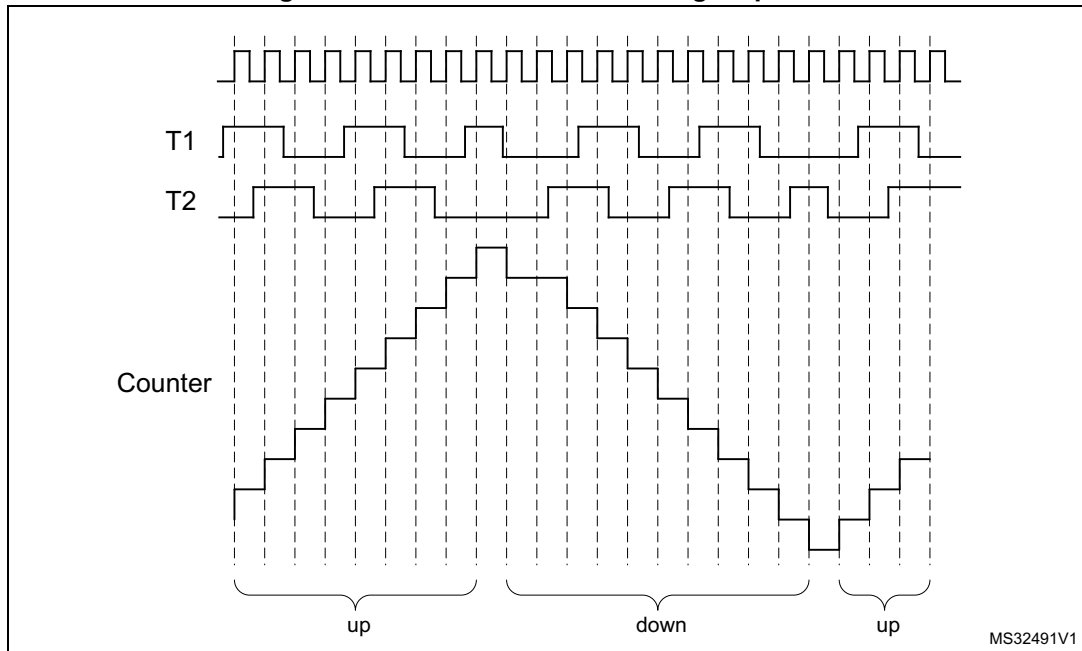
**Table 144. Encoder counting scenarios**

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

**Caution:** In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to '0'. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be '000').

Figure 277. Encoder mode counting sequence



### 26.4.16 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG\_LPTIM\_STOP configuration bit in the DBG module.

## 26.5 LPTIM low-power modes

Table 145. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Low-power run	No effect.
Low-power sleep	No effect. LPTIM interrupts cause the device to exit the Low-power sleep mode.
Stop 0 / Stop 1	No effect when LPTIM is clocked by LSE or LSI. LPTIM interrupts cause the device to exit Stop 0 and Stop 1.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	



## 26.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM\_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).

*Note: If any bit in the LPTIM\_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM\_ISR register (Status Register) is set, the interrupt is not asserted.*

**Table 146. Interrupt events**

Interrupt event	Description
Compare match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).
Auto-reload match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).
External trigger event	Interrupt flag is raised when an external trigger event is detected
Auto-reload register update OK	Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.
Compare register update OK	Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.
Direction change	Used in Encoder mode. Two interrupt flags are embedded to signal direction change: <ul style="list-style-type: none"> <li>– UP flag signals up-counting direction change</li> <li>– DOWN flag signals down-counting direction change.</li> </ul>

## 26.7 LPTIM registers

Refer to [Section 1.2: List of abbreviations for registers](#) for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

### 26.7.1 LPTIM interrupt and status register (LPTIM\_ISR)

Address offset: 0x000

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.	DOWN	UP	ARR OK	CMP OK	EXT TRIG	ARRM	CMPM
									r	r	r	r	r	r	r

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

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM\_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 26.3: LPTIM implementation.*

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM\_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 26.3: LPTIM implementation.*

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM\_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM\_ICR register.

Bit 3 **CMPOK**: Compare register update OK

CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM\_CMP register has been successfully completed. CMPOK flag can be cleared by writing 1 to the CMPOKCF bit in the LPTIM\_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM\_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM\_CNT register's value reached the LPTIM\_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM\_ICR register.

Bit 0 **CMPM**: Compare match

The CMPM bit is set by hardware to inform application that LPTIM\_CNT register value reached the LPTIM\_CMP register's value. CMPM flag can be cleared by writing 1 to the CMPMCF bit in the LPTIM\_ICR register.

### 26.7.2 LPTIM interrupt clear register (LPTIM\_ICR)

Address offset: 0x004

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.	DOWN CF	UPCF	ARRO KCF	CMPO KCF	EXTTR IGCF	ARRM CF	CMPM CF
									w	w	w	w	w	w	w

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

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM\_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 26.3: LPTIM implementation.*

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM\_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 26.3: LPTIM implementation.*

Bit 4 **ARROKCF**: Autoreload register update OK clear flag

Writing 1 to this bit clears the ARROK flag in the LPTIM\_ISR register

Bit 3 **CMPOKCF**: Compare register update OK clear flag

Writing 1 to this bit clears the CMPOK flag in the LPTIM\_ISR register

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag

Writing 1 to this bit clears the EXTTRIG flag in the LPTIM\_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag

Writing 1 to this bit clears the ARRM flag in the LPTIM\_ISR register

Bit 0 **CMPMCF**: Compare match clear flag

Writing 1 to this bit clears the CMPM flag in the LPTIM\_ISR register

### 26.7.3 LPTIM interrupt enable register (LPTIM\_IER)

Address offset: 0x008

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.	DOWN IE	UPIE	ARRO KIE	CMPO KIE	EXT TRIGIE	ARRM IE	CMPM IE
									rw	rw	rw	rw	rw	rw	rw

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

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable

- 0: DOWN interrupt disabled
- 1: DOWN interrupt enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to [Section 26.3: LPTIM implementation](#).

Bit 5 **UPIE**: Direction change to UP Interrupt Enable

- 0: UP interrupt disabled
- 1: UP interrupt enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to [Section 26.3: LPTIM implementation](#).

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable

- 0: ARROK interrupt disabled
- 1: ARROK interrupt enabled

Bit 3 **CMPOKIE**: Compare register update OK Interrupt Enable

- 0: CMPOK interrupt disabled
- 1: CMPOK interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable

- 0: EXTTRIG interrupt disabled
- 1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable

- 0: ARRM interrupt disabled
- 1: ARRM interrupt enabled

Bit 0 **CMPMIE**: Compare match Interrupt Enable

- 0: CMPM interrupt disabled
- 1: CMPM interrupt enabled

**Caution:** The LPTIM\_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0')

### 26.7.4 LPTIM configuration register (LPTIM\_CFGR)

Address offset: 0x00C

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.	ENC	COUNT MODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
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 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

- Bit 24 **ENC**: Encoder mode enable  
The ENC bit controls the Encoder mode  
0: Encoder mode disabled  
1: Encoder mode enabled  
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to [Section 26.3: LPTIM implementation](#).*
- Bit 23 **COUNTMODE**: counter mode enabled  
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:  
0: the counter is incremented following each internal clock pulse  
1: the counter is incremented following each valid clock pulse on the LPTIM external Input1
- Bit 22 **PRELOAD**: Registers update mode  
The PRELOAD bit controls the LPTIM\_ARR and the LPTIM\_CMP registers update modality  
0: Registers are updated after each APB bus write access  
1: Registers are updated at the end of the current LPTIM period
- Bit 21 **WAVPOL**: Waveform shape polarity  
The WAVEPOL bit controls the output polarity  
0: The LPTIM output reflects the compare results between LPTIM\_CNT and LPTIM\_CMP registers  
1: The LPTIM output reflects the inverse of the compare results between LPTIM\_CNT and LPTIM\_CMP registers
- Bit 20 **WAVE**: Waveform shape  
The WAVE bit controls the output shape  
0: Deactivate Set-once mode  
1: Activate the Set-once mode
- Bit 19 **TIMOUT**: Timeout enable  
The TIMOUT bit controls the Timeout feature  
0: A trigger event arriving when the timer is already started will be ignored  
1: A trigger event arriving when the timer is already started will reset and restart the counter
- Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity  
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:  
00: software trigger (counting start is initiated by software)  
01: rising edge is the active edge  
10: falling edge is the active edge  
11: both edges are active edges
- Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 8 available sources:

000: lptim\_ext\_trig0

001: lptim\_ext\_trig1

010: lptim\_ext\_trig2

011: lptim\_ext\_trig3

100: lptim\_ext\_trig4

101: lptim\_ext\_trig5

110: lptim\_ext\_trig6

111: lptim\_ext\_trig7

See [Section 26.4.3: LPTIM input and trigger mapping](#) for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]**: Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1

001: /2

010: /4

011: /8

100: /16

101: /32

110: /64

111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]**: Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger

01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.

10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.

11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]**: Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

- 00: any external clock signal level change is considered as a valid transition
- 01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
- 10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
- 11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]**: Clock polarity

If LPTIM is clocked by an external clock source:

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

- 00:the rising edge is the active edge used for counting.  
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.
- 01:the falling edge is the active edge used for counting  
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.
- 10:both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.  
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.
- 11:not allowed

Refer to [Section 26.4.15: Encoder mode](#) for more details about Encoder mode sub-modes.

Bit 0 **CKSEL**: Clock selector

The CKSEL bit selects which clock source the LPTIM will use:

- 0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)
- 1: LPTIM is clocked by an external clock source through the LPTIM external Input1

**Caution:** The LPTIM\_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').

### 26.7.5 LPTIM control register (LPTIM\_CR)

Address offset: 0x010

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.	RST ARE	COUN TRST	CNT STRT	SNG STRT	ENAB LE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE**: Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM\_CNT register will asynchronously reset LPTIM\_CNT register content.

This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST**: Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit will trigger a synchronous reset of the LPTIM\_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

**Caution:** COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT**: Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in Continuous mode. If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM\_ARR and LPTIM\_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 1 **SNGSTRT**: LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in single pulse mode. If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM\_ARR and LPTIM\_CNT registers.

This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 0 **ENABLE**: LPTIM enable

The ENABLE bit is set and cleared by software.

0:LPTIM is disabled

1:LPTIM is enabled



### 26.7.6 LPTIM compare register (LPTIM\_CMP)

Address offset: 0x014

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
CMP[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:0 **CMP[15:0]**: Compare value  
 CMP is the compare value used by the LPTIM.

**Caution:** The LPTIM\_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

### 26.7.7 LPTIM autoreload register (LPTIM\_ARR)

Address offset: 0x018

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.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bits 15:0 **ARR[15:0]**: Auto reload value  
 ARR is the autoreload value for the LPTIM.  
 This value must be strictly greater than the CMP[15:0] value.

**Caution:** The LPTIM\_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to '1').

### 26.7.8 LPTIM counter register (LPTIM\_CNT)

Address offset: 0x01C

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
CNT[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 **CNT[15:0]**: Counter value

When the LPTIM is running with an asynchronous clock, reading the LPTIM\_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

It should be noted that for a reliable LPTIM\_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.

### 26.7.9 LPTIM configuration register 2 (LPTIM\_CFGR2)

Address offset: 0x024

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.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

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

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection

The IN2SEL bits control the LPTIM Input 2 multiplexer, which connect LPTIM Input 2 to one of the available inputs.

00: lptim\_in2\_mux0

01: lptim\_in2\_mux1

10: lptim\_in2\_mux2

11: lptim\_in2\_mux3

For connection details refer to [Section 26.4.3: LPTIM input and trigger mapping](#).

*Note:* If the LPTIM does not support encoder mode feature, these bits are reserved. Please refer to [Section 26.3: LPTIM implementation](#).

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

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection

The IN1SEL bits control the LPTIM Input 1 multiplexer, which connects LPTIM Input 1 to one of the available inputs.

00: lptim\_in1\_mux0

01: lptim\_in1\_mux1

10: lptim\_in1\_mux2

11: lptim\_in1\_mux3

For connection details refer to [Section 26.4.3: LPTIM input and trigger mapping](#).

**Caution:** The LPTIM\_CFGR2 register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').

26.7.10 LPTIM register map

The following table summarizes the LPTIM registers.

Table 147. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x000	LPTIM_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.	DOWN <sup>(1)</sup>	UP <sup>(1)</sup>	AROK	CMPOK	EXTTRIG	ARRM	CMPM	
	Reset value																										0	0	0	0	0	0		
0x004	LPTIM_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.	DOWNCF <sup>(1)</sup>	UPCF <sup>(1)</sup>	AROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF	
	Reset value																										0	0	0	0	0	0	0	
0x008	LPTIM_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.	DOWNIE <sup>(1)</sup>	UPIE <sup>(1)</sup>	AROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE	
	Reset value																										0	0	0	0	0	0	0	
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC <sup>(1)</sup>	COUNTMODE	PRELOAD	WAVEPOL	WAVE	TIMOUT	TRIGEN	Res.	Res.	TRIGSEL[2:0]	Res.	Res.	Res.	Res.	PRESC	Res.	Res.	Res.	TRGFLT	Res.	CKFLT	CKPOL	CKSEL	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	
0x010	LPTIM_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.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE	
	Reset value																												0	0	0	0	0	
0x014	LPTIM_CMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x020	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.	Res.	Res.
0x024	LPTIM_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																	

1. If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to [Section 26.3: LPTIM implementation](#).

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 27 Infrared interface (IRTIM)

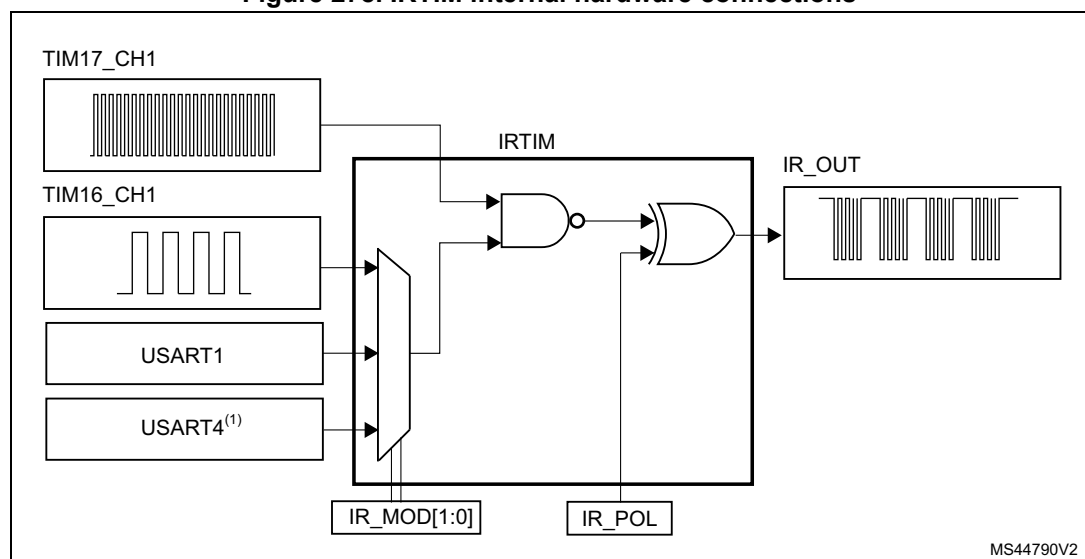
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with USART1, USART4 (on STM32G071xx/G081xx/G0B1xx/G0C1xx) or USART2 (on STM32G031xx/G041xx/G051xx/G061xx), TIM16, and TIM17 as shown in [Figure 278](#).

To generate the infrared remote control signals, the IR interface must be enabled and TIM16 channel 1 (TIM16\_OC1) and TIM17 channel 1 (TIM17\_OC1) must be properly configured to generate correct waveforms.

The infrared receiver can be implemented easily through a basic input capture mode.

**Figure 278. IRTIM internal hardware connections**



3. USART4 on STM32G071xx/G081xx/G0B1xx/G0C1xx, and USART2 on STM32G031xx/G041xx/G051xx/G061xx.

All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM17 is used to generate the high frequency carrier signal, while TIM16 or alternatively USART1 or USART4 generates the modulation envelope according to the setting of the IR\_MOD[1:0] bits in the SYSCFG\_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 pin) can be activated through the I2C\_PB9\_FMP bit in the SYSCFG\_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

## 28 Independent watchdog (IWDG)

### 28.1 Introduction

The devices feature an embedded watchdog peripheral (IWDG) that offers a combination of high safety level, timing accuracy, and flexibility of use. This peripheral detects and solves malfunctions due to software failure, and triggers a system reset when the counter reaches a given timeout value.

The independent watchdog is clocked by its own dedicated low-speed clock (LSI), and stays active even if the main clock fails.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. For further information on the window watchdog, refer to [Section 29: System window watchdog \(WWDG\)](#).

### 28.2 IWDG main features

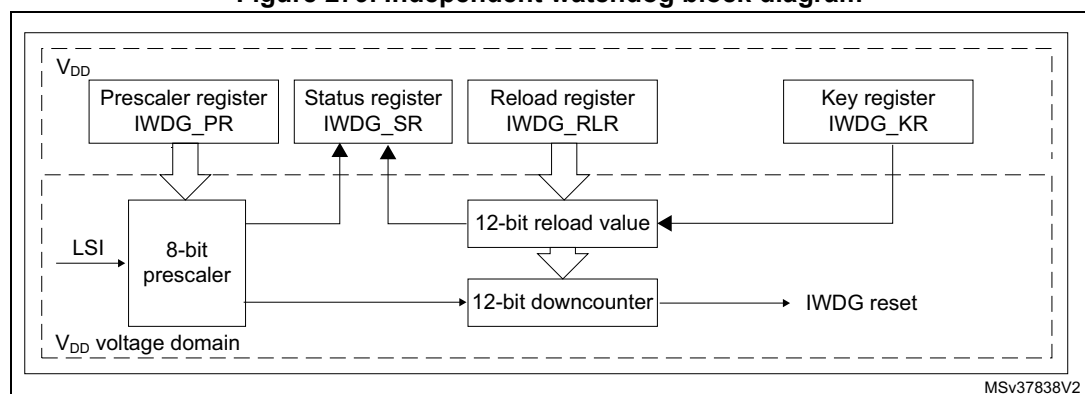
- Free-running downcounter
- Clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Conditional reset
  - Reset (if watchdog is activated) when the downcounter value becomes lower than 0x000
  - Reset (if watchdog is activated) if the downcounter is reloaded outside the window

### 28.3 IWDG functional description

#### 28.3.1 IWDG block diagram

[Figure 279](#) shows the functional blocks of the independent watchdog module.

**Figure 279. 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.

### 28.3.2 Window option

The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG\_WINR)*.

If the reload operation is performed while the counter is greater than the value stored in the *IWDG window register (IWDG\_WINR)*, a reset is provided.

The default value of the *IWDG window register (IWDG\_WINR)* is 0x0000 0FFF, so if it is not updated, the window option is disabled.

As soon as the window value is changed, a reload operation is performed to reset the downcounter to the *IWDG reload register (IWDG\_RLR)* value, and to ease the cycle number calculation to generate the next reload.

#### Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG\_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Write to the *IWDG window register (IWDG\_WINR)*. This automatically refreshes the counter value in the *IWDG reload register (IWDG\_RLR)*.

*Note:* Writing the window value allows the counter value to be refreshed by the RLR when the *IWDG status register (IWDG\_SR)* is set to 0x0000 0000.

#### Configuring the IWDG when the window option is disabled

When the window option is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG\_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG\_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG\_PR)* from 0 to 7.
4. Write the *IWDG reload register (IWDG\_RLR)*.
5. Wait for the registers to be updated (IWDG\_SR = 0x0000 0000).
6. Refresh the counter value with IWDG\_RLR (IWDG\_KR = 0x0000 AAAA).

### 28.3.3 Hardware watchdog

If this feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and generates a reset unless the *IWDG key register (IWDG\_KR)* is written by the software before the counter reaches the end of count, and if the downcounter is lower than the window value (WIN[11:0]).

### 28.3.4 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, 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.

### 28.3.5 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.



## 28.4 IWDG registers

Refer to [Section 1.2 on page 55](#) for a list of abbreviations used in register descriptions.  
 The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

### 28.4.1 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 28.3.4: Register access protection](#))

Writing the key value 0xCCCC starts the watchdog (except if the hardware watchdog option is selected)

### 28.4.2 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.	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.	PR[2:0]		
													rw	rw	rw

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

Bits 2:0 **PR[2:0]**: Prescaler divider

These bits are write access protected see [Section 28.3.4: 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.*

### 28.4.3 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<sub>DD</sub> voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on it. For this reason the value read from this register is valid only when the RVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

### 28.4.4 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.	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.	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 prescaled clock cycles).

Window value can be updated only when WVU bit is reset.

**Bit 1 RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V<sub>DD</sub> voltage domain (takes up to five prescaled clock cycles).

Reload value can be updated only when RVU bit is reset.

**Bit 0 PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V<sub>DD</sub> voltage domain (takes up to five LSI clock cycles).

Prescaler value can be updated only when PVU bit is reset.

*Note: If several reload, prescaler, or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, and to wait until WVU bit is reset before changing the window value. However, after updating the prescaler and/or the reload/window value it is not necessary to wait until RVU or PVU or WVU is reset before continuing code execution except in case of low-power mode entry.*

### 28.4.5 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 28.3.4](#), 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<sub>DD</sub> voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the [IWDG status register \(IWDG\\_SR\)](#) is reset.*

### 28.4.6 IWDG register map

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

**Table 148. IWDG 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	IWDG_KR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	KEY[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	IWDG_PR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR[2:0]			
	Reset value																																0	0	0	
0x08	IWDG_RLR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RL[11:0]													
	Reset value																							1	1	1	1	1	1	1	1	1	1	1		
0x0C	IWDG_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res		
	Reset value																																		0	0
0x10	IWDG_WINR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WIN[11:0]													
	Reset value																							1	1	1	1	1	1	1	1	1	1	1	1	

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

## 29 System window watchdog (WWDG)

### 29.1 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.

### 29.2 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 281](#))
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

### 29.3 WWDG functional description

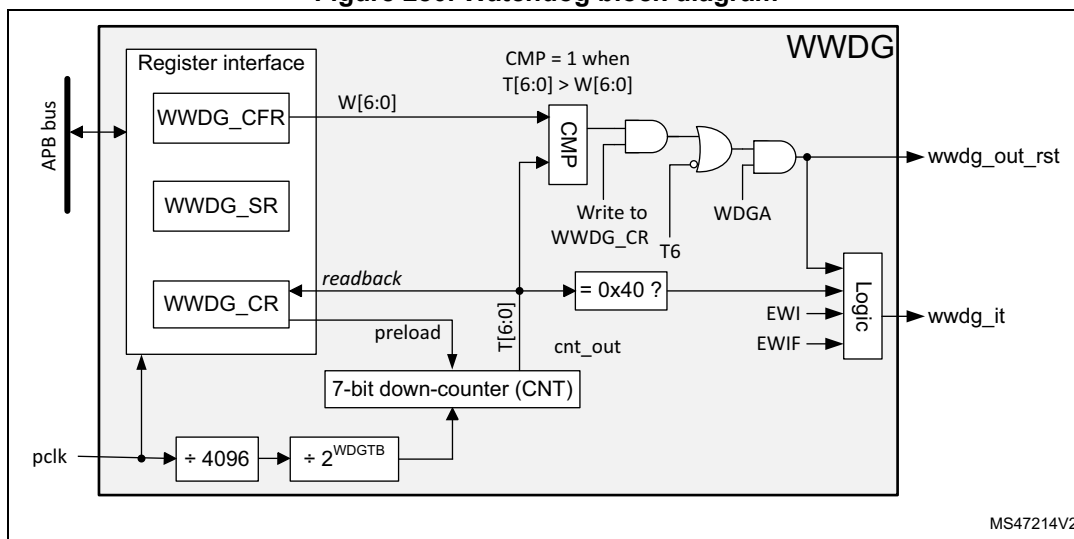
If the watchdog is activated (the WDGA bit is set in the WWDG\_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window 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 280](#) for the WWDG block diagram.

### 29.3.1 WWDG block diagram

Figure 280. Watchdog block diagram



### 29.3.2 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.

### 29.3.3 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 281](#)). 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 281](#) describes the window watchdog process.

*Note:* The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

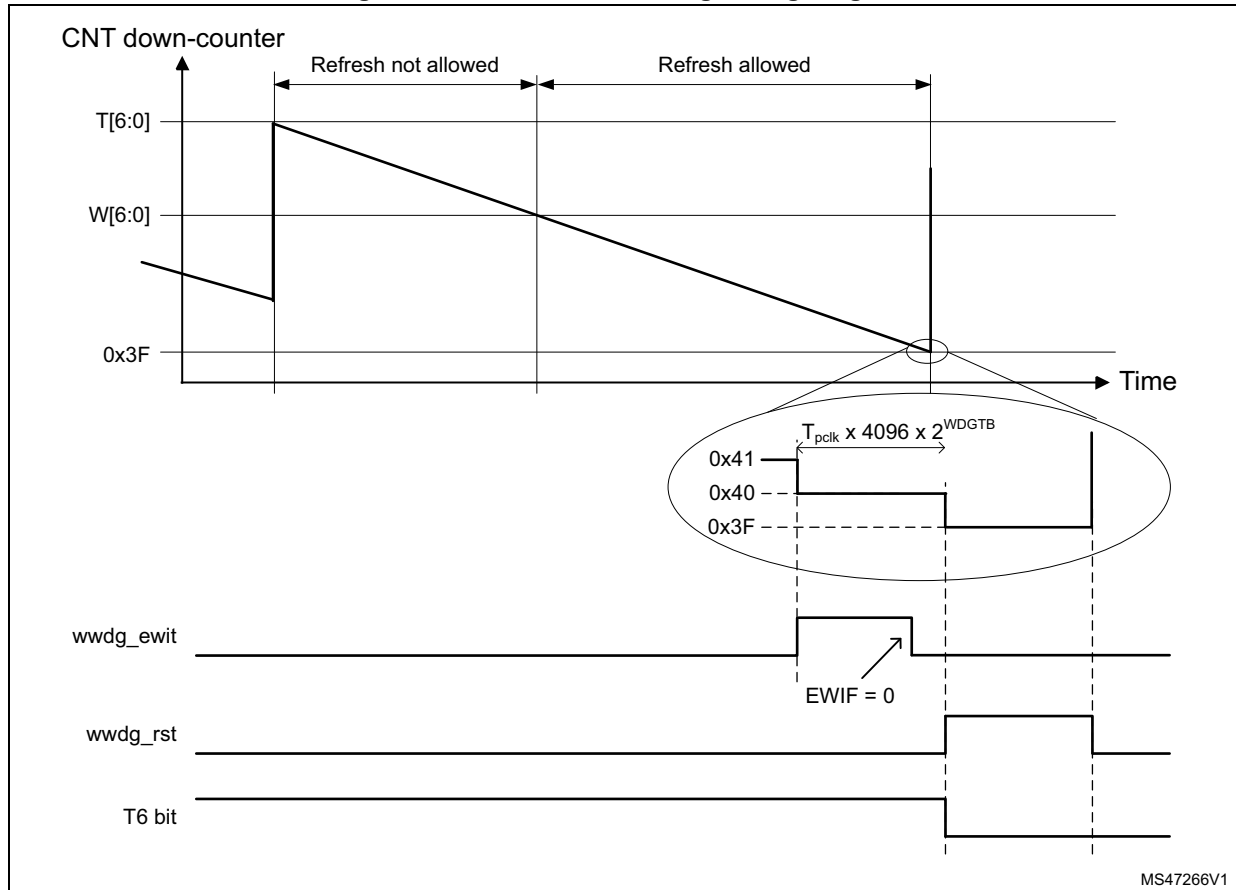
### 29.3.4 How to program the watchdog timeout

Use the formula in [Figure 281](#) to calculate the WWDG timeout.



**Warning:** When writing to the WWDG\_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 281. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (\text{ms})$$

where:

- $t_{WWDG}$ : WWDG timeout
- $t_{PCLK}$ : APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDG TB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1 / 48000) \times 4096 \times 2^3 \times (63 + 1) = 43.69\text{ms}$$

Refer to the datasheet for the minimum and maximum values of  $t_{WWDG}$ .

### 29.3.5 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 40: Debug support \(DBG\)](#).

## 29.4 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.*

## 29.5 WWDG registers

Refer to [Section 1.2: List of abbreviations for registers](#) for a list of abbreviations used in register descriptions.

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

### 29.5.1 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.	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.	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  $(4096 \times 2^{WDGTB[2:0]})$  PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

### 29.5.2 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
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

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.

### 29.5.3 WWDG status register (WWDG\_SR)

Address offset: 0x008

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.	Res.	Res.	Res.	EWIF
															rc_w0

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.

### 29.5.4 WWDG register map

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

**Table 149. WWDG register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x000	WWDG_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.	WDGA	T[6:0]							
	Reset value																									0	1	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWI	Res.	Res.	W[6:0]							
	Reset value																							0	0	0	1	1	1	1	1	1	1	1
0x008	WWDG_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.	Res.	EWIF
	Reset value																																	

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

## 30 Real-time clock (RTC)

### 30.1 Introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in  $V_{BAT}$  mode.

### 30.2 RTC main features

The RTC supports the following features (see [Figure 282: RTC block diagram](#)):

- Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to  $V_{BAT}$  mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.

The RTC is supplied through a switch that takes power either from the  $V_{DD}$  supply when present or from the  $V_{BAT}$  pin.

The RTC clock sources can be:

- A 32.768 kHz external crystal (LSE)
- An external resonator or oscillator (LSE)
- The internal low power RC oscillator (LSI, with typical frequency of 32 kHz)
- The high-speed external clock (HSE), divided by a prescaler in the RCC.

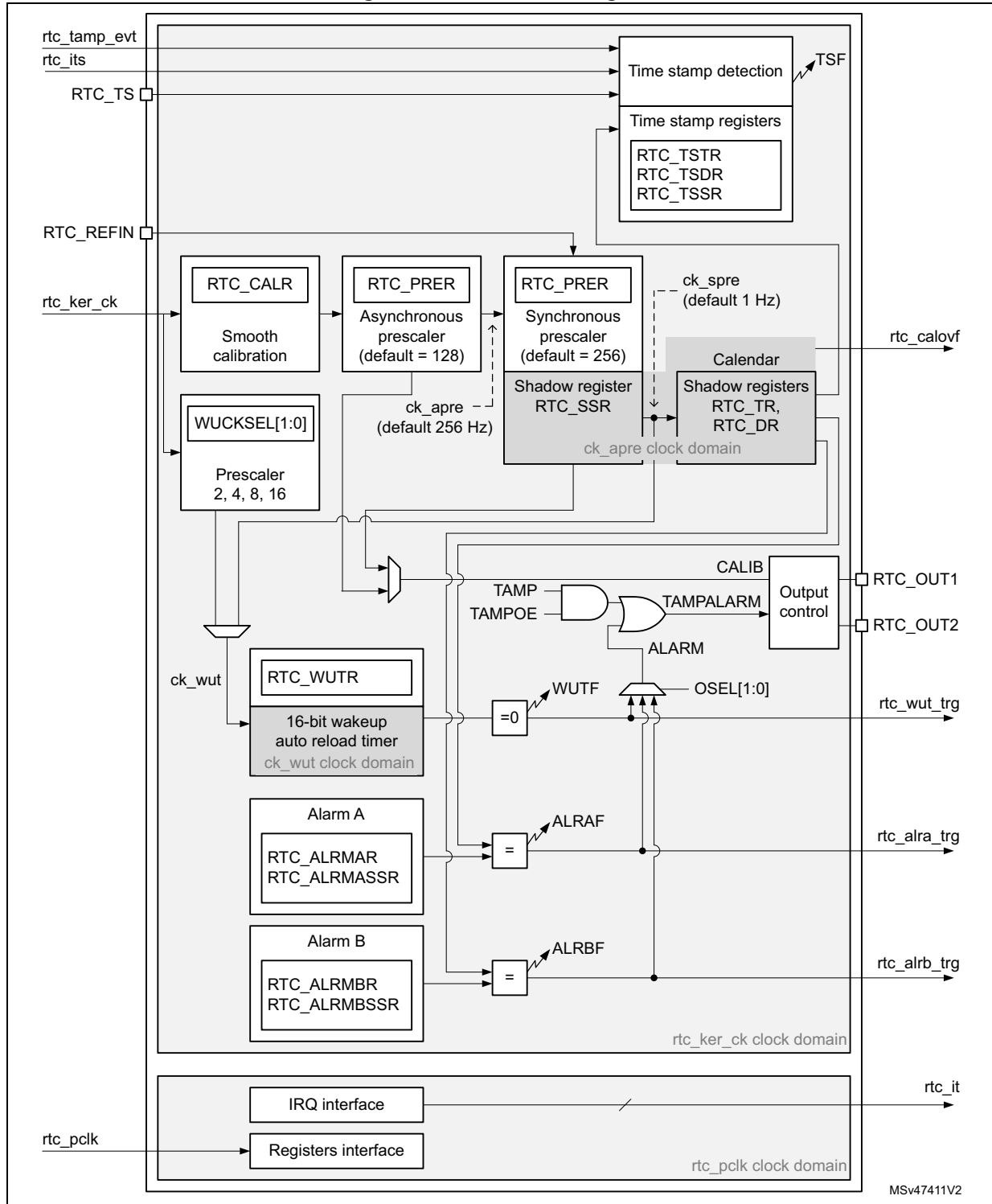
The RTC is functional in  $V_{BAT}$  mode and in all low-power modes when it is clocked by the LSE. When clocked by the LSI, the RTC is not functional in  $V_{BAT}$  mode, but is functional in all low-power modes except Shutdown mode.

All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

### 30.3 RTC functional description

#### 30.3.1 RTC block diagram

Figure 282. RTC block diagram



### 30.3.2 RTC pins and internal signals

**Table 150. 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 OR between TAMP and ALARM outputs.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC\_CR register which select the alarm A, alarm B or wake-up outputs. TAMP is enabled by setting the TAMPOE bit in the RTC\_CR register which selects the tamper event outputs.

**Table 151. 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_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_it	Output	RTC interrupts (refer to <a href="#">Section 30.5: RTC interrupts</a> for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or  $V_{BAT}$  when the selected clock is not LSE. Refer to [Section 30.4: RTC low-power modes](#) for more details.

**Table 152. RTC interconnection**

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V <sub>BAT</sub> detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

### 30.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the RTC domain are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

Both RTC and TAMP peripherals provide functions on these I/Os (refer to [Section 31: Tamper and backup registers \(TAMP\)](#)).

RTC\_OUT1, RTC\_TS and TAMP\_IN1 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V<sub>BAT</sub> mode.

The output mechanism follows the priority order shown in [Table 153](#).

**Table 153. PC13 configuration<sup>(1)</sup>**

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull	01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care
	00	1						
	01 or 10 or 11	1						



Table 153. PC13 configuration<sup>(1)</sup> (continued)

PC13 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Open-Drain <sup>(2)</sup>	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	1	0
		00	0	1	1				
		Don't care	Don't care	0					
RTC_TS and TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	1	1
		00	0	1	1				
		Don't care	Don't care	0					
RTC_TS input floating		00	0	0	Don't care	Don't care	Don't care	0	1
		00	0	1	1				
		Don't care	Don't care	0					

Table 153. PC13 configuration<sup>(1)</sup> (continued)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
Wake-up pin or Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0
	00	0	1	1				
	Don't care	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 on PA4 pin thanks to OUT2EN bit. This output is not available in V<sub>BAT</sub> mode. The different functions are mapped on RTC\_OUT1 or on RTC\_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in [Table 154](#).

Table 154. RTC\_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PA4
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

### 30.3.4 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

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 282: RTC block diagram](#)):

- A 7-bit asynchronous prescaler configured through the PREDIV\_A bits of the RTC\_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV\_S bits of the RTC\_PRER register.

*Note:* When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck\_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is  $2^{22}$ .

This corresponds to a maximum input frequency of around 4 MHz.

$f_{ck\_apre}$  is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck\_apre clock is used to clock the binary RTC\_SSR subseconds downcounter. When it reaches 0, RTC\_SSR is reloaded with the content of PREDIV\_S.

$f_{ck\_spre}$  is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck\_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see [Section 30.3.7: Periodic auto-wake-up](#) for details).

### 30.3.5 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 30.6.10: RTC shift control register \(RTC\\_SHIFTR\)](#)). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the

BYPHAD control bit in the RTC\_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC\_SSR, RTC\_TR or RTC\_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock ( $f_{APB}$ ) must be at least 7 times the frequency of the RTC clock ( $f_{RTCCLK}$ ).

The shadow registers are reset by system reset.

### 30.3.6 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC\_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC\_ALRMASR and RTC\_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC\_ALRMAR register, and through the MASKSSx bits of the RTC\_ALRMASR register.

The alarm interrupt is enabled through the ALRAIE bit in the RTC\_CR register.

**Caution:** If the seconds field is selected (MSK1 bit reset in RTC\_ALRMAR), the synchronous prescaler division factor set in the RTC\_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC\_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC\_CR register.

### 30.3.7 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC\_CR register.

The wake-up timer clock input ck\_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.  
When RTCCLK is LSE (32.768 kHz), this allows to configure the wake-up interrupt period from 122  $\mu$ s to 32 s, with a resolution down to 61  $\mu$ s.
- ck\_spre (usually 1 Hz internal clock)  
When ck\_spre frequency is 1 Hz, this allows to achieve a wake-up time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case  $2^{16}$  is added to the 16-bit counter current value. When the initialization sequence is complete (see [Programming the wake-up timer on page 874](#)), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag

is set in the RTC\_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC\_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC\_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC\_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC\_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wake-up timer.

### 30.3.8 RTC initialization and configuration

#### RTC register access

The RTC registers are 32-bit registers. The APB interface introduces two wait states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD = 0.

#### RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After RTC domain reset, some of the RTC registers are write-protected.

Writing to the protected RTC registers is enabled by writing a key into the Write Protection register, RTC\_WPR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC\_WPR register.
2. Write 0x53 into the RTC\_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

#### Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC\_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC\_ICSR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC\_PRER register.
4. Load the initial time and date values in the shadow registers (RTC\_TR and RTC\_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC\_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

*Note:* After a system reset, the application can read the INITS flag in the RTC\_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its RTC domain reset default value (0x00).  
To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC\_ICSR register.

### Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC\_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

### Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC\_CR to disable alarm A.
2. Program the alarm A registers (RTC\_ALRMASR/RTC\_ALRMAR).
3. Set ALRAE in the RTC\_CR register to enable alarm A again.

*Note:* Each change of the RTC\_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

### Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC\_WUTR):

1. Clear WUTE in RTC\_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC\_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wake-up auto-reload value WUT[15:0], and the wake-up clock selection (WUCKSEL[2:0] bits in RTC\_CR). Set WUTE in RTC\_CR to enable the timer again.

The wake-up timer restarts down-counting. The WUTWF bit is cleared up to 2 RTCCLK clocks cycles after WUTE is cleared, due to clock synchronization.

### 30.3.9 Reading the calendar

#### When BYPSHAD control bit is cleared in the RTC\_CR register

To read the RTC calendar registers (RTC\_SSR, RTC\_TR and RTC\_DR) properly, the APB1 clock frequency ( $f_{PCLK}$ ) must be equal to or greater than seven times the RTC clock frequency ( $f_{RTCCLK}$ ). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC\_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC\_ICSR register each time the calendar registers are copied into the RTC\_SSR, RTC\_TR and RTC\_DR shadow registers. The copy is performed every RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC\_SSR or RTC\_TR locks the values in the higher-order calendar shadow registers until RTC\_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to [Calendar initialization and configuration on page 873](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

After synchronization (refer to [Section 30.3.11: RTC synchronization](#)): the software must wait until RSF is set before reading the RTC\_SSR, RTC\_TR and RTC\_DR registers.

#### When the BYPSHAD control bit is set in the RTC\_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

**Note:** While `BYPSHAD = 1`, instructions which read the calendar registers require one extra APB cycle to complete.

### 30.3.10 Resetting the RTC

The calendar shadow registers (`RTC_SSR`, `RTC_TR` and `RTC_DR`) and some bits of the RTC status register (`RTC_ICSR`) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a RTC domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (`RTC_CR`), the prescaler register (`RTC_PREER`), the RTC calibration register (`RTC_CALR`), the RTC shift register (`RTC_SHIFTR`), the RTC timestamp registers (`RTC_TSSSR`, `RTC_TSTR` and `RTC_TSDR`), the wake-up timer register (`RTC_WUTR`), and the alarm A and alarm B registers (`RTC_ALRMASR/RTC_ALRMAR` and `RTC_ALRMBSSR/RTC_ALRMBR`).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the RTC domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a RTC domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

### 30.3.11 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (`RTC_SSR` or `RTC_TSSSR`), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using `RTC_SHIFTR`.

`RTC_SSR` contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of  $1 / (\text{PREDIV}_S + 1)$  seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (`PREDIV_S[14:0]`). The maximum resolution allowed (30.52  $\mu$ s with a 32768 Hz clock) is obtained with `PREDIV_S` set to `0x7FFF`.

However, increasing `PREDIV_S` means that `PREDIV_A` must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (`RTC_SHIFTR`). Writing to `RTC_SHIFTR` can shift (either delay or advance) the clock by up to a second with a resolution of  $1 / (\text{PREDIV}_S + 1)$  seconds. The shift operation consists of adding the `SUBFS[14:0]` value to the synchronous prescaler counter `SS[15:0]`: this will delay the clock. If at the same time the `ADD1S` bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

**Caution:** Before initiating a shift operation, the user must check that `SS[15] = 0` in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the `RTC_SHIFTR` register, the `SHPF` flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.



**Caution:** This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC\_SHIFTR when REFCKON = 1.

### 30.3.12 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC\_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC\_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC\_REFIN detection is enabled (REFCKON bit of RTC\_CR set to 1), the calendar is still clocked by the LSE, and RTC\_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC\_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck\_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck\_apre periods when detecting the first reference clock edge. A smaller window of 3 ck\_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck\_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck\_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck\_apre period detection window centered on the ck\_spre edge.

When the RTC\_REFIN detection is enabled, PREDIV\_A and PREDIV\_S must be set to their default values:

- PREDIV\_A = 0x007F
- PREDIV\_S = 0x00FF

*Note:* RTC\_REFIN clock detection is not available in Standby mode.

### 30.3.13 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about  $2^{20}$  RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal\_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC\_CALR) specifies the number of RTCCLK clock cycles to be masked during the 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 = 0xFF800).

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.

### 30.3.14 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC\_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC\_TSSSR, RTC\_TSTR, RTC\_TSDR) when a timestamp event is detected on the RTC\_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC\_TSSSR, RTC\_TSTR, RTC\_TSDR) when a tamper event is detected on the TAMP\_INx pinx.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC\_TSSSR, RTC\_TSTR, RTC\_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the  $V_{BAT}$  supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC\_SR register is set. In case the event is internal, the ITSF flag is also set in RTC\_SR register.

By setting the TSIE bit in the RTC\_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC\_TSTR and RTC\_TSDR) maintain the results of the previous event.

**Note:** *TSF is set 2  $ck_{apre}$  cycles after the timestamp event occurs due to synchronization process.*

*There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.*

**Caution:** If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in the RTC control register (RTC\_CR).

### 30.3.15 Calibration clock output

When the COE bit is set to 1 in the RTC\_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC\_CR register is reset and PREDIV\_A = 0x7F, the CALIB frequency is  $f_{RTCCLK}/64$ . 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 the CALIB output is selected, the RTC\_OUT1 or RTC\_OUT2 pin is automatically configured.*

*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.*

### 30.3.16 Tamper and alarm output

The OSEL[1:0] control bits in the RTC\_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC\_SR register.

When the TAMPOE control bit is set in the RTC\_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tamper flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC\_CR so that the opposite of the selected flags bit is output when POL is set to 1.

**TAMPALRM output**

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM\_TYPE in the RTC\_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM\_PU in the RTC\_CR.

*Note:* Once the TAMPALRM output is enabled, it has priority over CALIB on RTC\_OUT1. When the TAMPALRM output is selected, the RTC\_OUT1 or RTC\_OUT2 pin is automatically configured.

**30.4 RTC low-power modes**

**Table 155. 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 remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.
Shutdown	The RTC remains active when the RTC clock source is LSE. RTC interrupts cause the device to exit the Shutdown mode.

The table below summarizes the RTC pins and functions capability in all modes.

**Table 156. RTC pins functionality over modes**

Functions	Functional in all low-power modes except Standby and Shutdown modes	Functional in Standby and Shutdown mode	Functional in V <sub>BAT</sub> mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	Yes	No

### 30.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

**Table 157. 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 and Standby mode	Exit from Shutdown mode
RTC	Alarm A	ALRAF	ALRAIE	write 1 in CALRAF	Yes	Yes <sup>(3)</sup>	Yes <sup>(4)</sup>
	Alarm B	ALRBF	ALRBIE	write 1 in CALRBF	Yes	Yes <sup>(3)</sup>	Yes <sup>(4)</sup>
	Timestamp	TSF	TSIE	write 1 in CTSF	Yes	Yes <sup>(3)</sup>	Yes <sup>(4)</sup>
	Wake-up timer interrupt	WUTF	WUTIE	write 1 in CWUTF	Yes	Yes <sup>(3)</sup>	Yes <sup>(4)</sup>

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. Wake-up from Stop and Standby modes is possible only when the RTC clock source is LSE or LSI.
4. Wake-up from Shutdown modes is possible only when the RTC clock source is LSE.

### 30.6 RTC registers

Refer to [Section 1.2 on page 55](#) of the reference manual for a list of abbreviations used in register descriptions.

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

#### 30.6.1 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 873](#) and [Reading the calendar on page 875](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x00

RTC domain 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	HT[1:0]		HU[3:0]			
									r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w



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

### 30.6.2 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 873](#) and [Reading the calendar on page 875](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x04

RTC domain 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.

### 30.6.3 RTC sub second register (RTC\_SSR)

Address offset: 0x08

RTC domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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 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.

### 30.6.4 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 873](#).

Address offset: 0x0C

RTC domain reset value: 0x0000 0007

System reset value: 0bxxxx 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECAL PF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUTW F	ALRB WF	ALRAW F
								rw	r	rc_w0	r	r	r	r	r



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

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC\_CALR register, indicating that the RTC\_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to [Re-calibration on-the-fly](#).

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 (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (RTC domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC\_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

- Bit 2 **WUTWF**: Wake-up timer write flag  
 This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC\_CR.  
 It is cleared by hardware in initialization mode.  
 0: Wake-up timer configuration update not allowed except in initialization mode  
 1: Wake-up timer configuration update allowed
- Bit 1 **ALRBWF**: Alarm B write flag  
 This bit is set by hardware when alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC\_CR.  
 It is cleared by hardware in initialization mode.  
 0: Alarm B update not allowed  
 1: Alarm B update allowed
- Bit 0 **ALRAWF**: Alarm A write flag  
 This bit is set by hardware when alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC\_CR.  
 It is cleared by hardware in initialization mode.  
 0: Alarm A update not allowed  
 1: Alarm A update allowed

### 30.6.5 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 873](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x10

RTC domain reset value: 0x007F 00FF

System reset: not affected

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV\_A[6:0]**: Asynchronous prescaler factor  
 This is the asynchronous division factor:  
 $ck\_apre\ frequency = RTCCLK\ frequency / (PREDIV\_A + 1)$

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV\_S[14:0]**: Synchronous prescaler factor  
 This is the synchronous division factor:  
 $ck\_spre\ frequency = ck\_apre\ frequency / (PREDIV\_S + 1)$

### 30.6.6 RTC wake-up timer register (RTC\_WUTR)

This register can be written only when WUTWF is set to 1 in RTC\_ICSR.

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x14

RTC domain reset value: 0x0000 FFFF

System reset: not affected

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

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

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck\_wut cycles. The ck\_wut period is selected through WUCKSEL[2:0] bits of the RTC\_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 1) ck\_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCCLK/2) is forbidden.

### 30.6.7 RTC control register (RTC\_CR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x18

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2 EN	TAMP ALRM TYPE	TAMP ALRM PU	Res.	Res.	TAMP OE	TAMP TS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rW	rW	rW			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	WUTIE	ALRB IE	ALRA IE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYP SHAD	REFCK ON	TS EDGE	WUCKSEL[2:0]		
rW	rW	rW	rW	rW	rW	rW	rW		rW	rW	rW	rW	rW	rW	rW

- 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  $\neq$  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  $\neq$  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  $\neq$  00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC\_OUT2 and TAMPALRM is output on RTC\_OUT1.
- Bit 30 **TAMPALRM\_TYPE**: TAMPALRM output type  
0: TAMPALRM is push-pull output  
1: TAMPALRM is open-drain output
- Bit 29 **TAMPALRM\_PU**: TAMPALRM pull-up enable  
0: No pull-up is applied on TAMPALRM output  
1: A pull-up is applied on TAMPALRM output
- Bits 28:27 Reserved, must be kept at reset value.
- Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM  
0: The tamper flag is not routed on TAMPALRM  
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.
- Bit 25 **TAMPTS**: Activate timestamp on tamper detection event  
0: Tamper detection event does not cause a RTC timestamp to be saved  
1: Save RTC timestamp on tamper detection event  
TAMPTS is valid even if TSE = 0 in the RTC\_CR register. Timestamp flag is set after the tamper flags, therefore if TAMPTS and TSIE are set, it is recommended to disable the tamper interrupts in order to avoid servicing 2 interrupts.
- Bit 24 **ITSE**: timestamp on internal event enable  
0: internal event timestamp disabled  
1: internal event timestamp enabled
- Bit 23 **COE**: Calibration output enable  
This bit enables the CALIB output  
0: Calibration output disabled  
1: Calibration output enabled
- Bits 22:21 **OSEL[1:0]**: Output selection  
These bits are used to select the flag to be routed to TAMPALRM output.  
00: Output disabled  
01: Alarm A output enabled  
10: Alarm B output enabled  
11: Wake-up output enabled
- Bit 20 **POL**: Output polarity  
This bit is used to configure the polarity of TAMPALRM output.  
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).  
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

- Bit 19 **COSEL**: Calibration output selection  
When COE = 1, this bit selects which signal is output on CALIB.  
0: Calibration output is 512 Hz  
1: Calibration output is 1 Hz  
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV\_A = 127 and PREDIV\_S = 255). Refer to [Section 30.3.15: Calibration clock output](#).
- Bit 18 **BKP**: Backup  
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.
- Bit 17 **SUB1H**: Subtract 1 hour (winter time change)  
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.  
Setting this bit has no effect when current hour is 0.  
0: No effect  
1: Subtracts 1 hour to the current time. This can be used for winter time change.
- Bit 16 **ADD1H**: Add 1 hour (summer time change)  
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.  
0: No effect  
1: Adds 1 hour to the current time. This can be used for summer time change
- Bit 15 **TSIE**: Timestamp interrupt enable  
0: Timestamp interrupt disable  
1: Timestamp interrupt enable
- Bit 14 **WUTIE**: Wake-up timer interrupt enable  
0: Wake-up timer interrupt disabled  
1: Wake-up timer interrupt enabled
- Bit 13 **ALRBIE**: Alarm B interrupt enable  
0: Alarm B interrupt disable  
1: Alarm B interrupt enable
- Bit 12 **ALRAIE**: Alarm A interrupt enable  
0: Alarm A interrupt disabled  
1: Alarm A interrupt enabled
- Bit 11 **TSE**: timestamp enable  
0: timestamp disable  
1: timestamp enable
- Bit 10 **WUTE**: Wake-up timer enable  
0: Wake-up timer disabled  
1: Wake-up timer enabled  
*Note: When the wake-up timer is disabled, wait for WUTWF=1 before enabling it again.*
- Bit 9 **ALRBE**: Alarm B enable  
0: Alarm B disabled  
1: Alarm B enabled
- Bit 8 **ALRAE**: Alarm A enable  
0: Alarm A disabled  
1: Alarm A enabled
- Bit 7 Reserved, must be kept at reset value.

- Bit 6 **FMT**: Hour format
  - 0: 24 hour/day format
  - 1: AM/PM hour format
  
- Bit 5 **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 **WUCKSEL[2:0]**: ck\_wut wake-up clock selection
  - 000: RTC/16 clock is selected
  - 001: RTC/8 clock is selected
  - 010: RTC/4 clock is selected
  - 011: RTC/2 clock is selected
  - 10x: ck\_spre (usually 1 Hz) clock is selected
  - 11x: ck\_spre (usually 1 Hz) clock is selected and 2<sup>16</sup> is added to the WUT counter value

*Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC\_ICSR/INITF = 1). WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].*

*Bits 2 to 0 of this register can be written only when RTC\_CR WUTE bit = 0 and RTC\_ICSR WUTWF bit = 1.*

*It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.*

*ADD1H and SUB1H changes are effective in the next second.*

### 30.6.8 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.	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.	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.

### 30.6.9 RTC calibration register (RTC\_CALR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x28

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

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 30.3.13: RTC smooth digital calibration](#).

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

*Note:* CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to [Section 30.3.13: RTC smooth digital calibration](#).

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

*Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 30.3.13: RTC smooth digital calibration.*

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2<sup>20</sup> 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 30.3.13: RTC smooth digital calibration on page 877.

### 30.6.10 RTC shift control register (RTC\_SHIFTR)

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC\_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:

$$\text{Delay (seconds)} = \text{SUBFS} / (\text{PREDIV}_S + 1)$$

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

$$\text{Advance (seconds)} = (1 - (\text{SUBFS} / (\text{PREDIV}_S + 1)))$$

*Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.*



### 30.6.11 RTC timestamp time register (RTC\_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC\_SR. It is cleared when TSF bit is reset.

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]		SU[3:0]				
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation  
 0: AM or 24-hour format  
 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

### 30.6.12 RTC timestamp date register (RTC\_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC\_SR. It is cleared when TSF bit is reset.

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

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

### 30.6.13 RTC timestamp sub second register (RTC\_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC\_SR. It is cleared when the TSF bit is reset.

Address offset: 0x38

RTC domain reset value: 0x0000 0000

System reset: not affected

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

### 30.6.14 RTC alarm A register (RTC\_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC\_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x40

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]		SU[3:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask  
 0: Alarm A set if the date/day match  
 1: Date/day don't care in alarm A comparison

Bit 30 **WDSEL**: Week day selection  
 0: DU[3:0] represents the date units  
 1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask  
 0: Alarm A set if the hours match  
 1: Hours don't care in alarm A comparison

Bit 22 **PM**: AM/PM notation  
 0: AM or 24-hour format  
 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask  
 0: Alarm A set if the minutes match  
 1: Minutes don't care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask  
 0: Alarm A set if the seconds match  
 1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

### 30.6.15 RTC alarm A sub second register (RTC\_ALRMASR)

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 873](#).

Address offset: 0x44

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[14:1] are don't care in alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don't care in alarm A comparison. Only SS[1:0] are compared.

3: SS[14:3] are don't care in alarm A comparison. Only SS[2:0] are compared.

...

12: SS[14:12] are don't care in alarm A comparison. SS[11:0] are compared.

13: SS[14:13] are don't care in alarm A comparison. SS[12:0] are compared.

14: SS[14] is don't care in alarm A comparison. SS[13:0] are compared.

15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

*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.

### 30.6.16 RTC alarm B register (RTC\_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC\_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in [RTC register write protection on page 873](#).

Address offset: 0x48

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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 B date mask  
 0: Alarm B set if the date and day match  
 1: Date and day don't care in alarm B comparison

Bit 30 **WSEL**: Week day selection  
 0: DU[3:0] represents the date units  
 1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask  
 0: Alarm B set if the hours match  
 1: Hours don't care in alarm B comparison

Bit 22 **PM**: AM/PM notation  
 0: AM or 24-hour format  
 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask  
 0: Alarm B set if the minutes match  
 1: Minutes don't care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask  
 0: Alarm B set if the seconds match  
 1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

### 30.6.17 RTC alarm B sub second register (RTC\_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC\_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

Address offset: 0x4C

RTC domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

0x1: SS[14:1] are don't care in alarm B comparison. Only SS[0] is compared.

0x2: SS[14:2] are don't care in alarm B comparison. Only SS[1:0] are compared.

0x3: SS[14:3] are don't care in alarm B comparison. Only SS[2:0] are compared.

...

0xC: SS[14:12] are don't care in alarm B comparison. SS[11:0] are compared.

0xD: SS[14:13] are don't care in alarm B comparison. SS[12:0] are compared.

0xE: SS[14] is don't care in alarm B comparison. SS[13:0] are compared.

0xF: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

### 30.6.18 RTC status register (RTC\_SR)

Address offset: 0x50

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
										r	r	r	r	r	r

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

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs. If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0. This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC\_TR and RTC\_DR) match the alarm B register (RTC\_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC\_TR and RTC\_DR) match the alarm A register (RTC\_ALRMAR).

*Note:* The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC\_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

### 30.6.19 RTC masked interrupt status register (RTC\_MISR)

Address offset: 0x54

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITS MF	TSOV MF	TS MF	WUT MF	ALRB MF	ALRA MF
										r	r	r	r	r	r

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

Bit 5 **ITSMF**: Internal timestamp masked flag  
 This flag is set by hardware when a timestamp on the internal event occurs and timestamp interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow masked flag  
 This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp masked flag  
 This flag is set by hardware when a timestamp interrupt occurs. If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer masked flag  
 This flag is set by hardware when the wake-up timer interrupt occurs. This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B masked flag  
 This flag is set by hardware when the alarm B interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag  
 This flag is set by hardware when the alarm A interrupt occurs.

*Note:* The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC\_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

### 30.6.20 RTC status clear register (RTC\_SCR)

Address offset: 0x5C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITSF	CTSOVF	CTS	CWUT	CALRBF	CALRAF
										F	F	F	F	F	F
										w	w	w	w	w	w

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

Bit 5 **CITSF**: Clear internal timestamp flag  
 Writing 1 in this bit clears the ITSF bit in the RTC\_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag  
 Writing 1 in this bit clears the TSOVF bit in the RTC\_SR register. It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.



Bit 3 **CTSF**: Clear timestamp flag

Writing 1 in this bit clears the TSOVF bit in the RTC\_SR register.

If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wake-up timer flag

Writing 1 in this bit clears the WUTF bit in the RTC\_SR register.

Bit 1 **CALRBF**: Clear alarm B flag

Writing 1 in this bit clears the ALRBF bit in the RTC\_SR register.

Bit 0 **CALRAF**: Clear alarm A flag

Writing 1 in this bit clears the ALRAF bit in the RTC\_SR register.

30.6.21 RTC register map

Table 158. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]	HU[3:0]			Res.	MNT[2:0]		MNU[3:0]			Res.	ST[2:0]		SU[3:0]									
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]			YU[3:0]			WDU[2:0]		MT	MU[3:0]			Res.	Res.	DT [1:0]	DU[3:0]								
	Reset value										0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1		0	0	0	0	1	
0x08	RTC_SSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUTF	ALBWF	ALRAWF
	Reset value																0										0	0	0	0	0	1	1	1
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						PREDIV_S[14:0]																	
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0x14	RTC_WUTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]																
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0x18	RTC_CR	OUTZEN	TAMPALRM_TYPE	TAMPALRM_PU	Res.	Res.	TAMPOE	TAMPPTS	ITSE	COE	SF [1:0]	POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPHAD	REFCKON	TSEDEG	WUCK SEL[2:0]				
	Reset value	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]								
	Reset value																									0	0	0	0	0	0	0	0	
0x28	RTC_CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]									
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0	
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]															
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]	HU[3:0]			Res.	MNT[2:0]		MNU[3:0]			Res.	ST[2:0]		SU[3:0]									
	Reset value										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]	MT	MU[3:0]			Res.	Res.	DT [1:0]	DU[3:0]								
	Reset value																	0	0	0	0	0	0	0		0	0	0	0	0	0	0		
0x38	RTC_TSSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	



Table 158. 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	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]	DU[3:0]			MSK3	PM	HT [1:0]	HU[3:0]			MSK2	MNT[2:0]		MNU[3:0]			MSK1	ST[2:0]		SU[3:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ALRMASR	Res.	Res.	Res.	Res.	MASKSS [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]															
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]	DU[3:0]			MSK3	PM	HT [1:0]	HU[3:0]			MSK2	MNT[2:0]		MNU[3:0]			MSK1	ST[2:0]		SU[3:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ALRMBSSR	Res.	Res.	Res.	Res.	MASKSS [3:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]															
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_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.	Res.	Res.
	Reset value																																	
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																																	
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 60](#) for the register boundary addresses.

## 31 Tamper and backup registers (TAMP)

### 31.1 Introduction

5 32-bit backup registers are retained in all low-power modes and also in  $V_{BAT}$  mode. They can be used to store sensitive data as their content is protected by an tamper detection circuit. Up to 3 tamper pins and 4 internal tampers are available for anti-tamper detection. The external tamper pins can be configured for edge detection, or level detection with or without filtering.

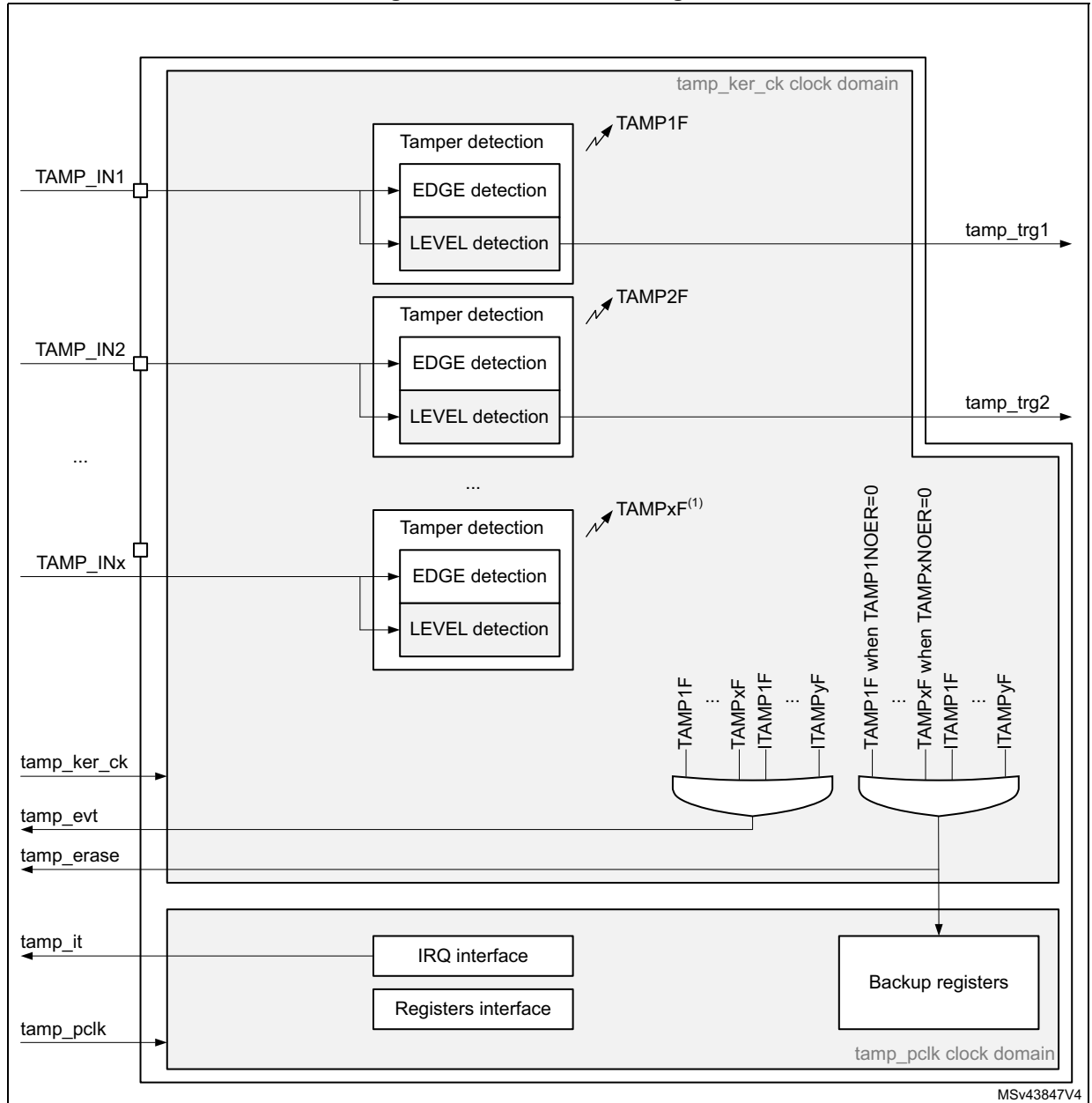
### 31.2 TAMP main features

- 5 backup registers:
  - the backup registers (TAMP\_BKPxR) are implemented in the RTC domain that remains powered-on by  $V_{BAT}$  when the  $V_{DD}$  power is switched off.
- Up to 3 external tamper detection events.
  - External passive tampers with configurable filter and internal pull-up.
- 4 internal tamper events.
- Any tamper detection can generate a RTC timestamp event.
- Any tamper detection can erase the backup registers.

### 31.3 TAMP functional description

#### 31.3.1 TAMP block diagram

Figure 283. TAMP block diagram



1. The number of external and internal tampers depends on products.

### 31.3.2 TAMP pins and internal signals

**Table 159. TAMP input/output pins**

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin

**Table 160. TAMP internal input/output signals**

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_evt	Output	Tamper event detection (internal or external) The tamp_evt is used to generate a RTC timestamp event
tamp_erase	Output	Device secrets erase request following tamper event detection (internal or external)
tamp_it	Output	TAMP interrupt (refer to <a href="#">Section 31.5: TAMP interrupts</a> for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or V<sub>BAT</sub> when the selected clock is not LSE (refer to [Section 31.4: TAMP low-power modes](#) for more details).

**Table 161. TAMP interconnection**

Signal name	Source/Destination
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_erase	The tamp_erase signal is used to erase the device secrets listed hereafter: backup registers
tamp_itamp3	LSE monitoring
tamp_itamp4	HSE monitoring
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	ST manufacturer readout

### 31.3.3 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

### 31.3.4 Tamper detection

The tamper detection can be configured for the following purposes:

- erase the backup registers (default configuration)
- generate an interrupt, capable to wake-up from Stop and Standby mode
- generate a hardware trigger for the low-power timers

#### TAMP backup registers

The backup registers (TAMP\_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs except if the TAMPxNOER bit is set, or if the TAMPxMSK is set in the TAMP\_CR2 register.

*Note: The backup registers are also erased when the readout protection of the flash is changed from level 1 to level 0.*

#### Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP\_CR register.

Each TAMP\_INx tamper detection input is associated with a flag TAMPxF in the TAMP\_SR register.

When TAMPxMSK is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck\_apre cycles when TAMPFLT differs from 0x0 (level detection with filtering)
- 3 ck\_apre cycles when TAMPTS = 1 (timestamp on tamper event)
- No latency when TAMPFLT = 0x0 (edge detection) and TAMPTS = 0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMSK is set:

A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck\_rtc additional cycles.

By setting the TAMPxIE bit in the TAMP\_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

#### Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit in cleared in TAMP\_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP\_SR register. This configuration allows to trig automatically the low-power timers in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the *Level detection with filtering on tamper inputs (passive mode)* mode (TAMPFLT ≠ 00 and active mode is not selected).

### Timestamp on tamper event

With TAMPTS set to 1 in the RTC\_CR, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit is set in RTC\_SR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set in the TAMP\_SR at the same time that TSF or TSOVF is set in the RTC\_SR.

### Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP\_INx pins generate tamper detection events when either a rising edge/high level or a falling edge/low level is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP\_INx inputs are deactivated when edge detection is selected.

**Caution:** When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection. When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP\_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP\_BKPxR). This prevents the application from writing to the backup registers while the TAMP\_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP\_INx input.

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

### Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP\_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP\_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

**Note:** *Refer to the datasheet for the electrical characteristics of the pull-up resistors.*



## 31.4 TAMP low-power modes

**Table 162. Effect of low-power modes on TAMP**

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby mode.
Shutdown	No effect on all features, except for level detection with filtering mode which remain active only when the clock source is LSE. TAMP interrupts cause the device to exit the Shutdown mode.

## 31.5 TAMP interrupts

The interrupt channel is set in the interrupt status register. The interrupt output is also activated.

**Table 163. 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 and Standby modes	Exit from Shutdown mode
TAMP	Tamper x <sup>(3)</sup>	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes	Yes <sup>(4)</sup>	Yes <sup>(5)</sup>
	Internal tamper y <sup>(3)</sup>	ITAMPyF	ITAMPyIE	Write 1 in CITAMPxF	Yes	Yes <sup>(4)</sup>	Yes <sup>(5)</sup>

1. The event flags are in the TAMP\_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the TAMP\_MISR register.
3. The number of tampers and internal tampers events depend on products.
4. In case of level detection with filtering passive tamper mode, wake-up from Stop and Standby modes is possible only when the TAMP clock source is LSE or LSI.
5. In case of level detection with filtering passive tamper mode, wake-up from Shutdown modes is possible only when the TAMP clock source is LSE.

## 31.6 TAMP registers

Refer to [Section 1.2 on page 55](#) of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

### 31.6.1 TAMP control register 1 (TAMP\_CR1)

Address offset: 0x00

RTC domain reset value: 0xFFFF 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.	ITAMP6 E	ITAMP5 E	ITAMP4 E	ITAMP3 E	Res.	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.	TAMP3 E	TAMP2 E	TAMP1 E
													rw	rw	rw

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

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ITAMP6E**: Internal tamper 6 enable: ST manufacturer readout

0: Internal tamper 6 disabled.

1: Internal tamper 6 enabled: a tamper is generated in case of ST manufacturer readout.

Bit 20 **ITAMP5E**: Internal tamper 5 enable: RTC calendar overflow

0: Internal tamper 5 disabled.

1: Internal tamper 5 enabled: a tamper is generated when the RTC calendar reaches its maximum value, on the 31<sup>st</sup> of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

Bit 19 **ITAMP4E**: Internal tamper 4 enable: HSE monitoring

0: Internal tamper 4 disabled.

1: Internal tamper 4 enabled. a tamper is generated when the HSE frequency is below or above thresholds.

Bit 18 **ITAMP3E**: Internal tamper 3 enable: LSE monitoring

0: Internal tamper 3 disabled.

1: Internal tamper 3 enabled: a tamper is generated when the LSE frequency is below or above thresholds.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3E**: Tamper detection on TAMP\_IN3 enable<sup>(1)</sup>

- 0: Tamper detection on TAMP\_IN3 is disabled.
- 1: Tamper detection on TAMP\_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP\_IN2 enable<sup>(1)</sup>

- 0: Tamper detection on TAMP\_IN2 is disabled.
- 1: Tamper detection on TAMP\_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP\_IN1 enable<sup>(1)</sup>

- 0: Tamper detection on TAMP\_IN1 is disabled.
- 1: Tamper detection on TAMP\_IN1 is enabled.

1. Tamper detection mode (selected with TAMP\_FLTCR register and TAMPxTRG bits in TAMP\_CR2), must be configured before enabling the tamper detection.

### 31.6.2 TAMP control register 2 (TAMP\_CR2)

Address offset: 0x04

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	Res.	Res.	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
					rw	rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3 NOER	TAMP2 NOER	TAMP1 NOER
													rw	rw	rw

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

Bit 26 **TAMP3TRG**: Active level for tamper 3 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 Tamper 3 input staying low triggers a tamper detection event.  
If TAMPFLT = 00 Tamper 3 input rising edge and high level triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 Tamper 3 input staying high triggers a tamper detection event.  
If TAMPFLT = 00 Tamper 3 input falling edge and low level triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 Tamper 2 input staying low triggers a tamper detection event.  
If TAMPFLT = 00 Tamper 2 input rising edge and high level triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 Tamper 2 input staying high triggers a tamper detection event.  
If TAMPFLT = 00 Tamper 2 input falling edge and low level triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 Tamper 1 input staying low triggers a tamper detection event.  
If TAMPFLT = 00 Tamper 1 input rising edge and high level triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 Tamper 1 input staying high triggers a tamper detection event.  
If TAMPFLT = 00 Tamper 1 input falling edge and low level triggers a tamper detection event.

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers are not erased.

*The tamper 3 interrupt must not be enabled when TAMP3MSK is set.*

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers are not erased.

*The tamper 2 interrupt must not be enabled when TAMP2MSK is set.*

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers are not erased.

*The tamper 1 interrupt must not be enabled when TAMP1MSK is set.*

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3NOER**: Tamper 3 no erase

0: Tamper 3 event erases the backup registers.

1: Tamper 3 event does not erase the backup registers.

Bit 1 **TAMP2NOER**: Tamper 2 no erase

0: Tamper 2 event erases the backup registers.

1: Tamper 2 event does not erase the backup registers.

Bit 0 **TAMP1NOER**: Tamper 1 no erase

0: Tamper 1 event erases the backup registers.

1: Tamper 1 event does not erase the backup registers.

### 31.6.3 TAMP filter control register (TAMP\_FLTCR)

Address offset: 0x0C

RTC domain reset value: 0x0000 0000

System reset: not affected

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

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

Bit 7 **TAMPPUDIS**: TAMP\_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.

0: Precharge TAMP\_INx pins before sampling (enable internal pull-up)

1: Disable precharge of TAMP\_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP\_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP\_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP\_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP\*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP\_INx inputs.

0x0: Tamper event is activated on edge of TAMP\_INx input transitions to the active level (no internal pull-up on TAMP\_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP\_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

*Note:* This register concerns only the tamper inputs in passive mode.

### 31.6.4 TAMP interrupt enable register (TAMP\_IER)

Address offset: 0x2C

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP6 IE	ITAMP5 IE	ITAMP4 IE	ITAMP3 IE	Res.	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.	TAMP 3IE	TAMP 2IE	TAMP 1IE
													rw	rw	rw

- Bits 31:24 Reserved, must be kept at reset value.
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 Reserved, must be kept at reset value.
- Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable: ST manufacturer readout  
 0: Internal tamper 6 interrupt disabled.  
 1: Internal tamper 6 interrupt enabled.
- Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable: RTC calendar overflow  
 0: Internal tamper 5 interrupt disabled.  
 1: Internal tamper 5 interrupt enabled.
- Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable: HSE monitoring  
 0: Internal tamper 4 interrupt disabled.  
 1: Internal tamper 4 interrupt enabled.
- Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable: LSE monitoring  
 0: Internal tamper 3 interrupt disabled.  
 1: Internal tamper 3 interrupt enabled.
- Bit 17 Reserved, must be kept at reset value.
- Bit 16 Reserved, must be kept at reset value.
- Bits 15:3 Reserved, must be kept at reset value.
- Bit 2 **TAMP3IE**: Tamper 3 interrupt enable  
 0: Tamper 3 interrupt disabled.  
 1: Tamper 3 interrupt enabled..
- Bit 1 **TAMP2IE**: Tamper 2 interrupt enable  
 0: Tamper 2 interrupt disabled.  
 1: Tamper 2 interrupt enabled.
- Bit 0 **TAMP1IE**: Tamper 1 interrupt enable  
 0: Tamper 1 interrupt disabled.  
 1: Tamper 1 interrupt enabled.

### 31.6.5 TAMP status register (TAMP\_SR)

Address offset: 0x30

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP6 F	ITAMP5 F	ITAMP4 F	ITAMP3 F	Res.	Res.
										r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP 3F	TAMP 2F	TAMP 1F
													r	r	r

- Bits 31:24 Reserved, must be kept at reset value.
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 Reserved, must be kept at reset value.
- Bit 21 **ITAMP6F**: ST manufacturer readout tamper detection flag  
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.
- Bit 20 **ITAMP5F**: RTC calendar overflow tamper detection flag  
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.
- Bit 19 **ITAMP4F**: HSE monitoring tamper detection flag  
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.
- Bit 18 **ITAMP3F**: LSE monitoring tamper detection flag  
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.
- Bit 17 Reserved, must be kept at reset value.
- Bit 16 Reserved, must be kept at reset value.
- Bits 15:3 Reserved, must be kept at reset value.
- Bit 2 **TAMP3F**: TAMP3 detection flag  
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.
- Bit 1 **TAMP2F**: TAMP2 detection flag  
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.
- Bit 0 **TAMP1F**: TAMP1 detection flag  
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

### 31.6.6 TAMP masked interrupt status register (TAMP\_MISR)

Address offset: 0x34

RTC domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP6 MF	ITAMP5 MF	ITAMP4 MF	ITAMP3 MF	Res.	Res.
										r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP 3MF	TAMP 2MF	TAMP 1MF
													r	r	r

- Bits 31:24 Reserved, must be kept at reset value.
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 Reserved, must be kept at reset value.



- Bit 21 **ITAMP6MF**: ST manufacturer readout tamper interrupt masked flag  
This flag is set by hardware when the internal tamper 6 interrupt is raised.
- Bit 20 **ITAMP5MF**: RTC calendar overflow tamper interrupt masked flag  
This flag is set by hardware when the internal tamper 5 interrupt is raised.
- Bit 19 **ITAMP4MF**: HSE monitoring tamper interrupt masked flag  
This flag is set by hardware when the internal tamper 4 interrupt is raised.
- Bit 18 **ITAMP3MF**: LSE monitoring tamper interrupt masked flag  
This flag is set by hardware when the internal tamper 3 interrupt is raised.
- Bit 17 Reserved, must be kept at reset value.
- Bit 16 Reserved, must be kept at reset value.
- Bits 15:3 Reserved, must be kept at reset value.
- Bit 2 **TAMP3MF**: TAMP3 interrupt masked flag  
This flag is set by hardware when the tamper 3 interrupt is raised.
- Bit 1 **TAMP2MF**: TAMP2 interrupt masked flag  
This flag is set by hardware when the tamper 2 interrupt is raised.
- Bit 0 **TAMP1MF**: TAMP1 interrupt masked flag  
This flag is set by hardware when the tamper 1 interrupt is raised.

### 31.6.7 TAMP status clear register (TAMP\_SCR)

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.	C ITAMP 6F	C ITAMP 5F	C ITAMP 4F	C ITAMP 3F	Res.	Res.
										w	w	w	w		
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.	CTAMP 3F	CTAMP 2F	CTAMP 1F
													w	w	w

- Bits 31:24 Reserved, must be kept at reset value.
- Bit 23 Reserved, must be kept at reset value.
- Bit 22 Reserved, must be kept at reset value.
- Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag  
Writing 1 in this bit clears the ITAMP6F bit in the TAMP\_SR register.
- Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag  
Writing 1 in this bit clears the ITAMP5F bit in the TAMP\_SR register.
- Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag  
Writing 1 in this bit clears the ITAMP4F bit in the TAMP\_SR register.
- Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag  
Writing 1 in this bit clears the ITAMP3F bit in the TAMP\_SR register.



Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag

Writing 1 in this bit clears the TAMP3F bit in the TAMP\_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag

Writing 1 in this bit clears the TAMP2F bit in the TAMP\_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag

Writing 1 in this bit clears the TAMP1F bit in the TAMP\_SR register.

### 31.6.8 TAMP backup x register (TAMP\_BKPxR)

Address offset:  $0x100 + 0x04 * x$ , ( $x = 0$  to  $4$ )

RTC domain reset value:  $0x0000\ 0000$

System reset: not affected

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

Bits 31:0 **BKP[31:0]**

The application can write or read data to and from these registers.

They are powered-on by  $V_{BAT}$  when  $V_{DD}$  is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.

In the default configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the readout protection (RDP) is disabled.

31.6.9 TAMP register map

Table 164. TAMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	TAMP_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3E	TAMP2E	TAMP1E	
	Reset value											1	1	1	1																0	0	0	
0x04	TAMP_CR2	Res.	Res.	Res.	Res.	Res.	TAMP3TRG	TAMP2TRG	TAMP1TRG	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3NOER	TAMP2NOER	TAMP1NOER
	Reset value						0	0	0							0	0	0													0	0	0	
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPUDIS	TAMPPRCH[1:0]	Res.	Res.	TAMPFLT[1:0]	Res.	Res.	TAMPFREQ[2:0]	
	Reset value																									0	0	0	0	0	0	0	0	
0x2C	TAMP_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value											0	0	0	0																0	0	0	
0x30	TAMP_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F
	Reset value											0	0	0	0																0	0	0	
0x34	TAMP_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value											0	0	0	0																0	0	0	
0x3C	TAMP_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP3F	CITAMP2F	CITAMP1F
	Reset value											0	0	0	0																0	0	0	
0x100 + 0x04*x, (x = 0 to 4)	TAMP_BKPxR	BKP[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.



## 32 Inter-integrated circuit interface (I2C)

### 32.1 Introduction

The I2C peripheral handles the interface between the device and the serial I<sup>2</sup>C (inter-integrated circuit) bus. It provides multicontroller 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+).

The I2C peripheral is also SMBus (system management bus) and PMBus<sup>®</sup> (power management bus) compatible.

It can use DMA to reduce the CPU load.

### 32.2 I2C main features

- I<sup>2</sup>C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller 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 target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility<sup>(a)</sup>:
  - 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. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to [Section 32.3: I2C implementation](#).

### 32.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

**Table 165. I2C implementation**

I2C features <sup>(1)</sup>	I2C1	I2C2	I2C3 <sup>(2)</sup>
7-bit addressing mode	X	X	X
10-bit addressing mode	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X
Independent clock	X	X <sup>(2)</sup> / -	-
Wake-up from Stop mode	X	X <sup>(2)</sup> / -	-
SMBus/PMBus	X	X <sup>(2)</sup> / -	-

1. X = supported.

2. Available on / applies to STM32G0B1xx and STM32G0C1xx devices only.

### 32.4 I2C functional description

In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I<sup>2</sup>C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I<sup>2</sup>C-bus.

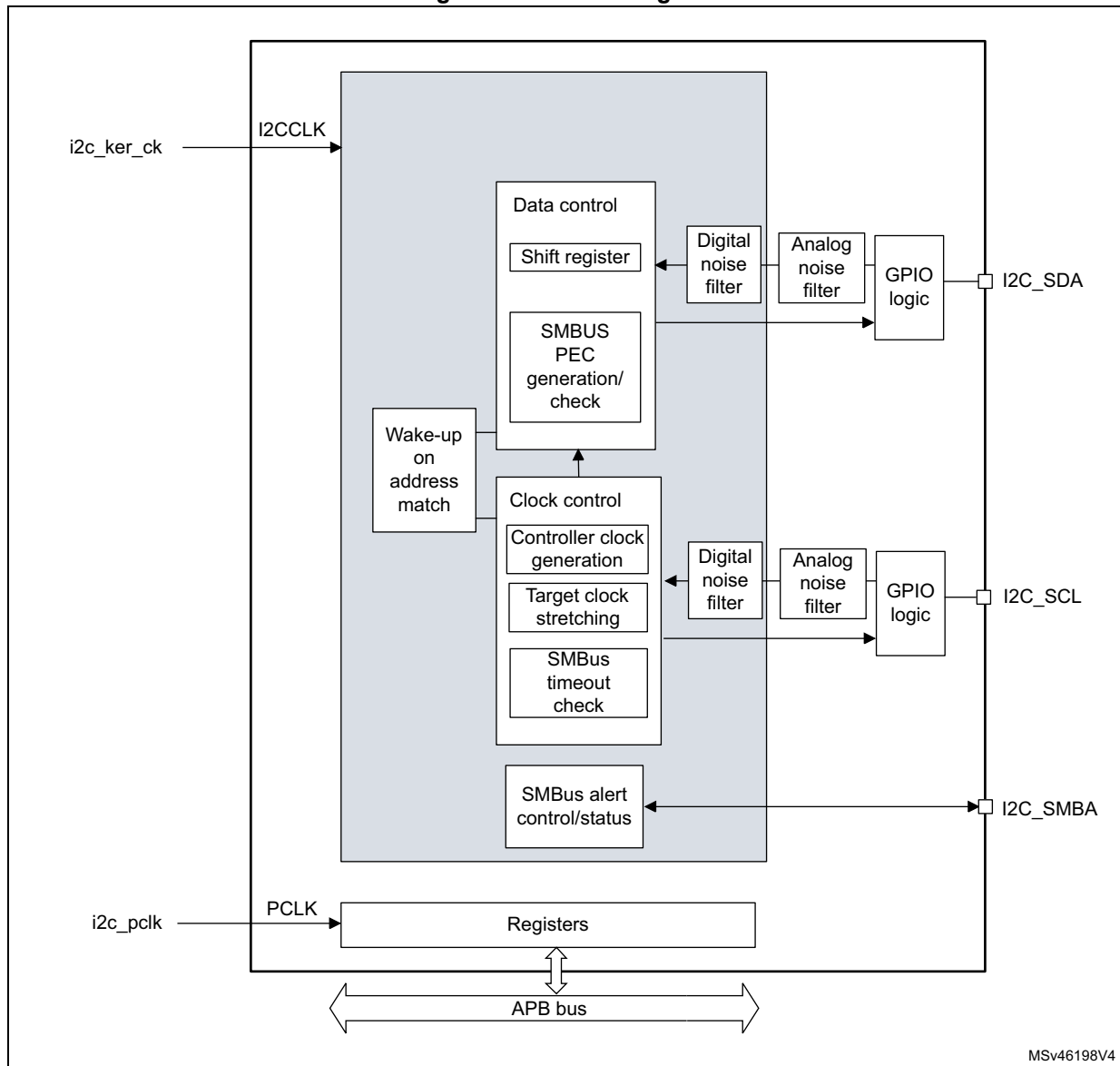
The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA). Refer to [Section 32.3: I2C implementation](#) for the relevant I2C instances.

The independent clock function allows the I2C communication speed to be independent of the PCLK frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration block(SYSCFG).

32.4.1 I2C block diagram

Figure 284. Block diagram



The block diagram shows a functional superset. Refer to [Section 32.3: I2C implementation](#) for information relative to different instances of the I2C peripheral. The instances not supporting features such as SMBus and wake-up from Stop mode do not include the corresponding blocks. The instances not supporting the independent clock function use PCLK for clocking both the kernel and the registers.

### 32.4.2 I2C pins and internal signals

Table 166. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I <sup>2</sup> C-bus data
I2C_SCL	Bidirectional	I <sup>2</sup> C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 167. 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 181</a> for the 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)

### 32.4.3 I2C clock requirements

The I2C kernel is clocked by I2CCLK.

The I2CCLK period  $t_{I2CCLK}$  must respect the following conditions:

$$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$$

$$t_{I2CCLK} < t_{HIGH}$$

where  $t_{LOW}$  is the SCL low time,  $t_{HIGH}$  is the SCL high time, and  $t_{filters}$  is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is  $DNF[3:0] \times t_{I2CCLK}$ .

The PCLK clock period  $t_{PCLK}$  must respect the condition  $t_{PCLK} < 4/3 t_{SCL}$ , where  $t_{SCL}$  is the SCL period.

**Caution:** When the I2C kernel is clocked by PCLK, this clock must respect the conditions for  $t_{I2CCLK}$ .

### 32.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I<sup>2</sup>C-bus environment.

**Communication flow**

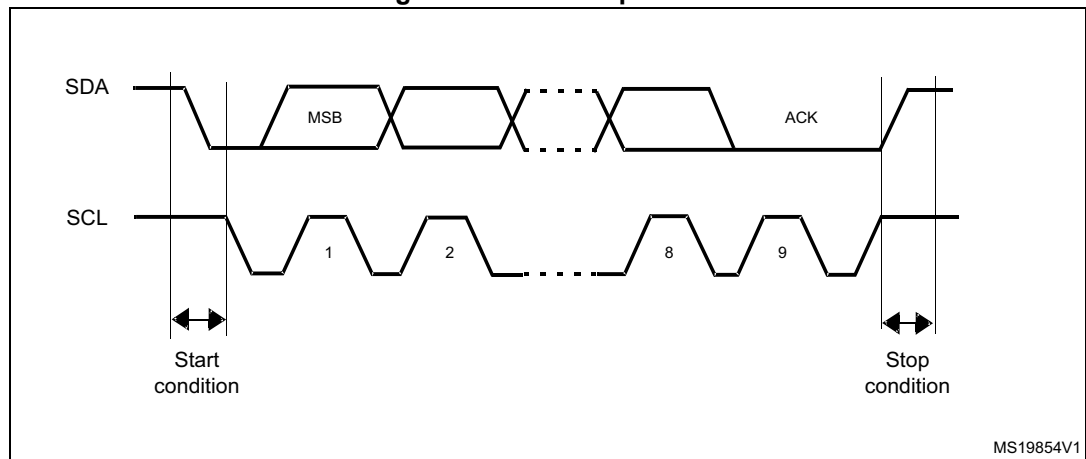
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

**Figure 285. I<sup>2</sup>C-bus protocol**



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

**32.4.5 I2C initialization**

**Enabling and disabling the peripheral**

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C\_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to [Section 32.4.6](#) for more details.

**Noise filters**

Before enabling the I2C peripheral by setting the PE bit of the I2C\_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I<sup>2</sup>C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C\_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I<sup>2</sup>C-bus line when remaining stable for more than DNF[3:0] periods of I2CCLK. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen I2CCLK periods.

The following table compares the two filters.

**Table 168. Comparison of analog and digital filters**

Item	Analog filter	Digital filter
Filtering capacity <sup>(1)</sup>	≥ 50 ns	One to fifteen I2CCLK periods
Benefits	Available in Stop mode	<ul style="list-style-type: none"> <li>– Programmable filtering capacity</li> <li>– Extra filtering capability versus I<sup>2</sup>C-bus specification requirements</li> <li>– Stable filtering capacity</li> </ul>
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

**Caution:** The filter configuration cannot be changed when the I2C peripheral is enabled.

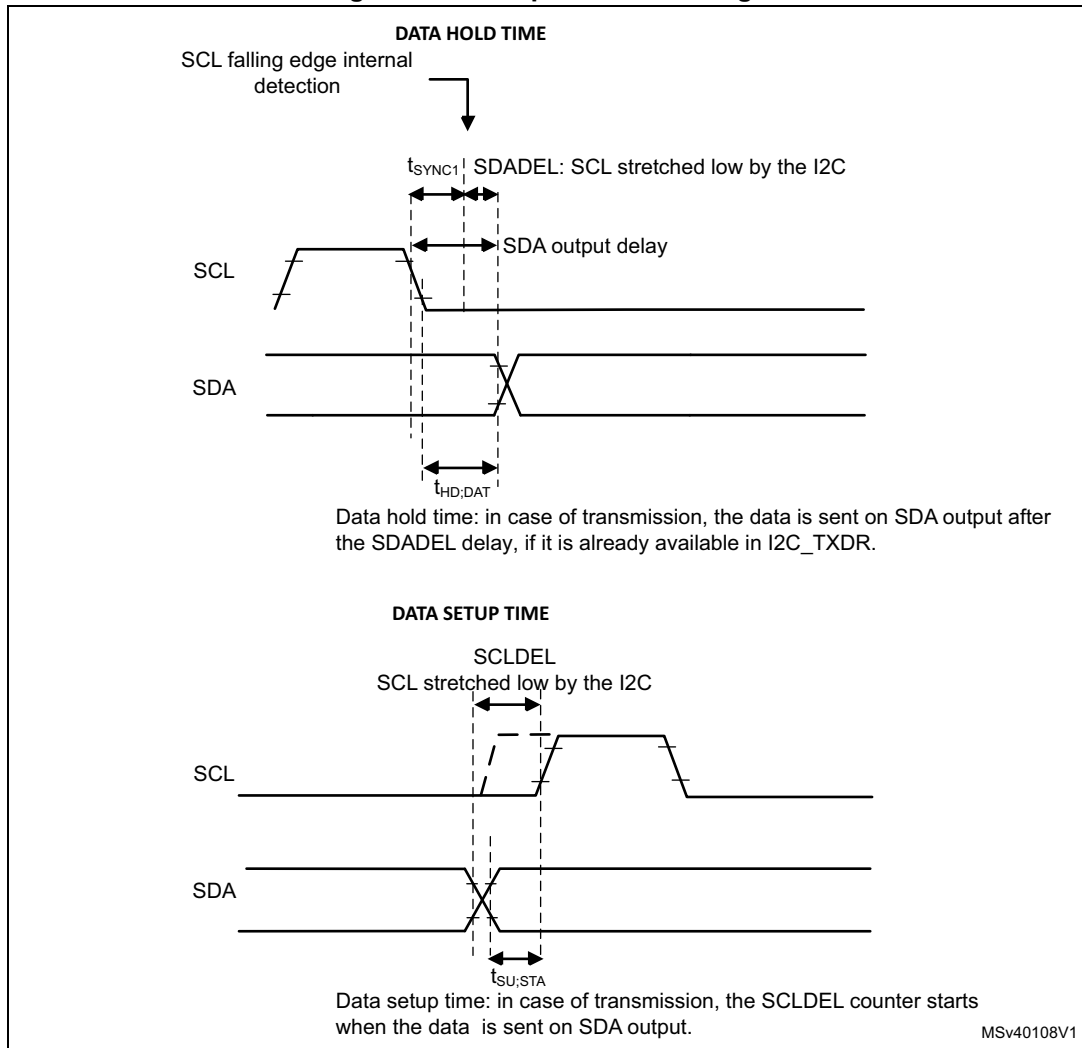
**I2C timings**

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C\_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the *I2C configuration* window.



Figure 286. Setup and hold timings



When the SCL falling edge is internally detected, the delay  $t_{SDADEL}$  (impacting the hold time  $t_{HD;DAT}$ ) is inserted before sending SDA output:

$$t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}, \text{ where } t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}.$$

The total SDA output delay is:

$$t_{SYNC1} + \{[SDADEL \times (PRESC + 1) + 1] \times t_{I2CCLK}\}$$

The  $t_{SYNC1}$  duration depends upon:

- SCL falling slope
- input delay  $t_{AF(min)} < t_{AF} < t_{AF(max)}$  introduced by the analog filter (if enabled)
- input delay  $t_{DNF} = DNF \times t_{I2CCLK}$  introduced by the digital filter (if enabled)
- delay due to SCL synchronization to I2CCLK clock (two to three I2CCLK periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

$$\{t_{i(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$$

$$SDADEL \leq \{t_{HD;DAT(max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

*Note:*  $t_{AF(min)}$  and  $t_{AF(max)}$  are only part of the condition when the analog filter is enabled. Refer to the device datasheet for  $t_{AF}$  values.

The  $t_{HD;DAT}$  time can at maximum be 3.45  $\mu$ s for Standard-mode, 0.9  $\mu$ s for Fast-mode, and 0.45  $\mu$ s for 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. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$$SDADEL \leq \{t_{VD;DAT(max)} - t_r(max) - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

*Note:* This condition can be violated when  $NOSTRETCH = 0$ , because the device stretches SCL low to guarantee the set-up time, according to the  $SCLDEL[3:0]$  value.

After  $t_{SDADEL}$ , or after sending SDA output when the target had to stretch the clock because the data was not yet written in  $I2C\_TXDR$  register, the SCL line is kept at low level during the setup time. This setup time is  $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$ , 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  $SCLDEL[3:0]$  so as to fulfill the following condition:

$$\{[t_r(max) + t_{SU;DAT(min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to the following table for  $t_f$ ,  $t_r$ ,  $t_{HD;DAT}$ ,  $t_{VD;DAT}$ , and  $t_{SU;DAT}$  standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed  $SDADEL[3:0]$  and  $SCLDEL[3:0]$  values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

*Note:* At every clock pulse, after SCL falling edge detection, I2C operating as controller or target 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 SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and  $SCLDEL$  counter starts, continuing stretching SCL low to guarantee the data setup time.

When the  $NOSTRETCH$  bit is set in target mode, the SCL is not stretched. The  $SDADEL[3:0]$  must then be programmed so that it ensures a sufficient setup time.

**Table 169. I<sup>2</sup>C-bus and 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	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	$\mu$ s
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
$t_r$	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
$t_f$	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C\_TIMINGR register.

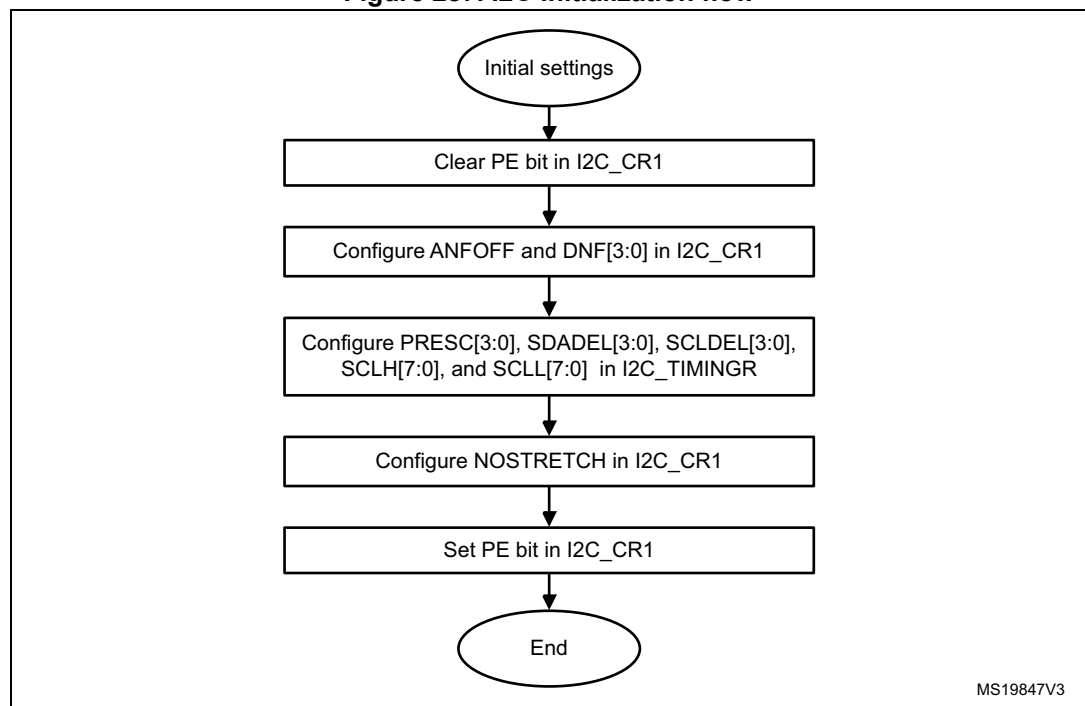
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay  $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$ , where  $t_{PRESC} = (PRESC + 1) \times t_{I2CCCLK}$ . The  $t_{SCLL}$  delay impacts the SCL low time  $t_{LOW}$ .

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay  $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$ , where  $t_{PRESC} = (PRESC + 1) \times t_{I2CCCLK}$ . The  $t_{SCLH}$  impacts the SCL high time  $t_{HIGH}$ .

Refer to [I2C controller initialization](#) for more details.

**Caution:** Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to [I2C target initialization](#) for more details.

Figure 287. I2C initialization flow



### 32.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C\_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C\_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C\_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

*Note:* The *PECBYTE*, *PECERR*, *TIMOUT*, and *ALERT* bits only apply to I2C instances supporting SMBus.

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

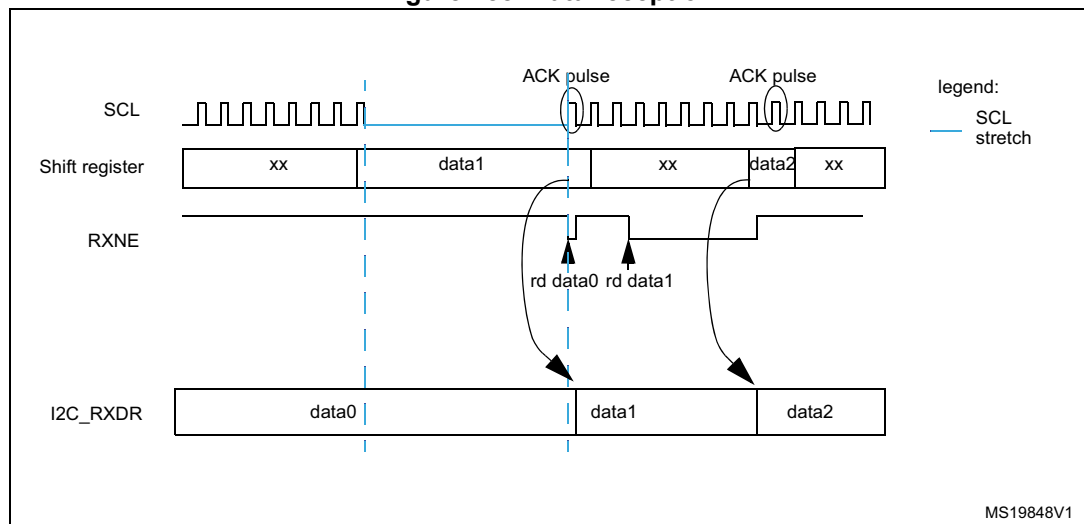
### 32.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

#### Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C\_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C\_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

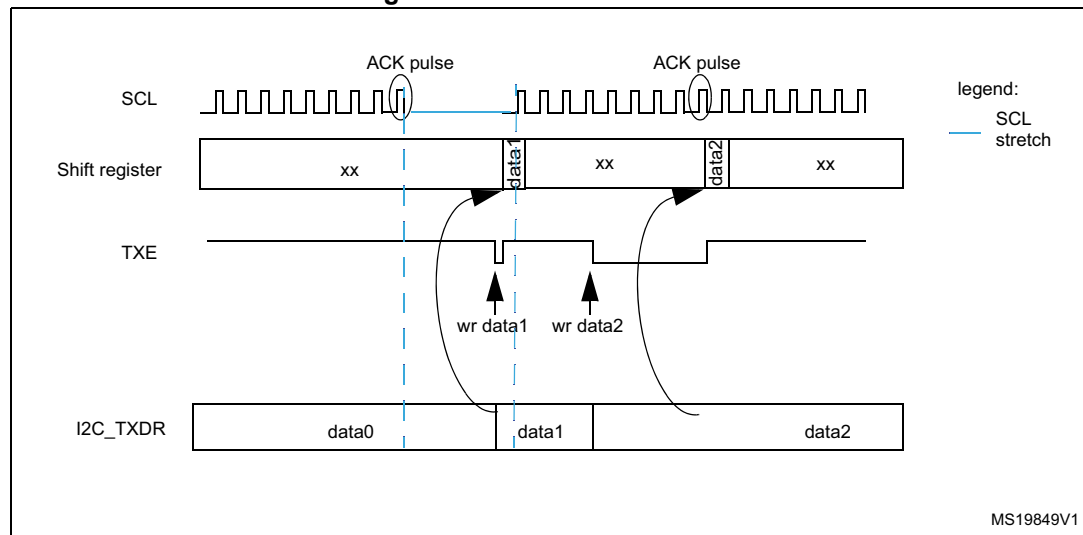
**Figure 288. Data reception**



## Transmission

If the I2C\_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C\_TXDR, the SCL line is stretched low until I2C\_TXDR is written. The stretch starts after the ninth SCL pulse.

**Figure 289. Data transmission**



## Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking, on I2C instances supporting SMBus

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C\_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C\_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C\_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C\_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C\_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C\_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

**Table 170. I2C configuration**

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

### 32.4.8 I2C target mode

#### I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C\_OAR1 and I2C\_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C\_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C\_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C\_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C\_OAR1 or I2C\_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C\_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C\_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C\_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C\_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

### Target with clock stretching

As long as the NOSTRETCH bit of the I2C\_CR1 register is zero (default), the I2C peripheral operating as an I<sup>2</sup>C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.  
The stretch is released when the software clears the ADDR flag by setting the ADDRCONF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C\_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).  
The stretch is released when the data is written to the I2C\_TXDR register.
- In reception, the I2C\_RXDR register is not read yet and a new data reception is completed.  
The stretch is released when I2C\_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).  
The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.  
The stretch is released after  $[(SDADEL + SCLDEL + 1) \times (PRESC + 1) + 1] \times t_{I2CCCLK}$  period.

### Target without clock stretching

As long as the NOSTRETCH bit of the I2C\_CR1 register is set, the I2C peripheral operating as an I<sup>2</sup>C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C\_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C\_ISR register and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C\_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C\_ISR register, and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

### Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C\_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C\_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C\_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

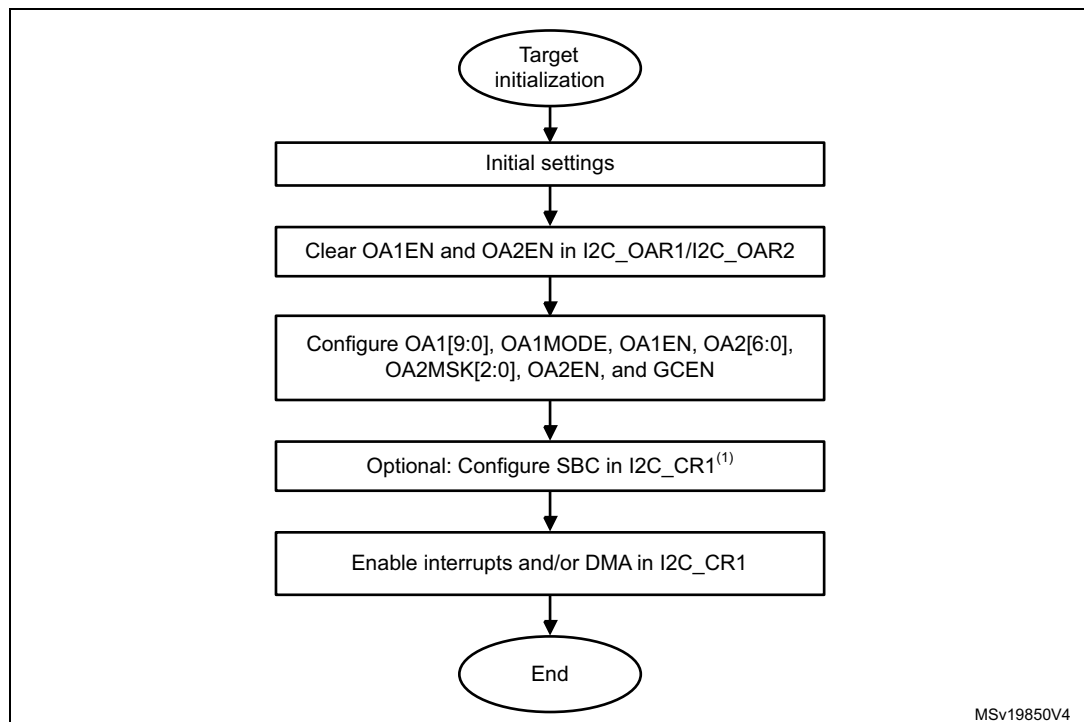
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

**Note:** The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

**Caution:** The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 290. Target initialization flow



MSv19850V4

1. SBC must be set to support SMBus features.



### Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C\_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C\_CR1 register is set.

The TXIS flag is cleared when the I2C\_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C\_ISR register and an interrupt is generated if the NACKIE bit of the I2C\_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C\_CR1 register is set, the STOPF flag of the I2C\_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target 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 target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

**Caution:** When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C\_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C\_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C\_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 291. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

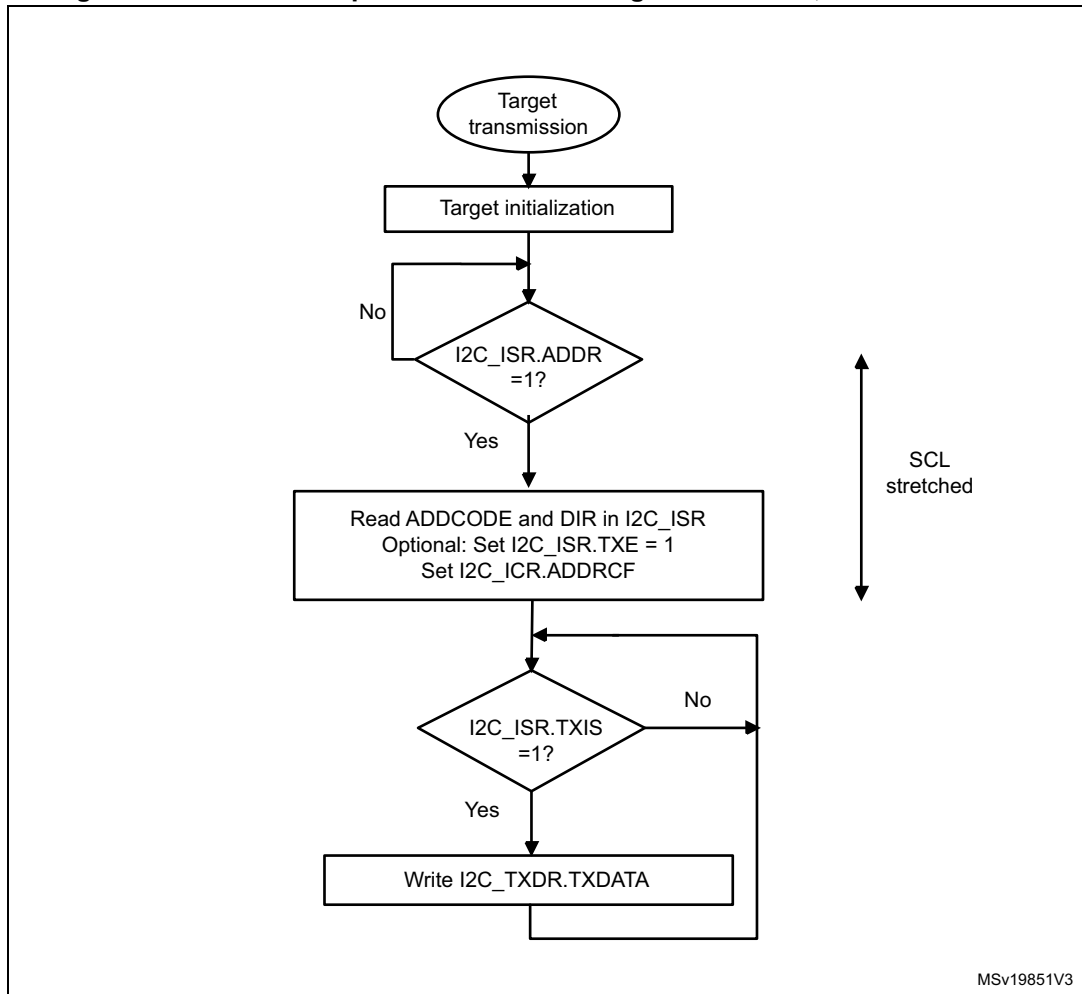


Figure 292. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

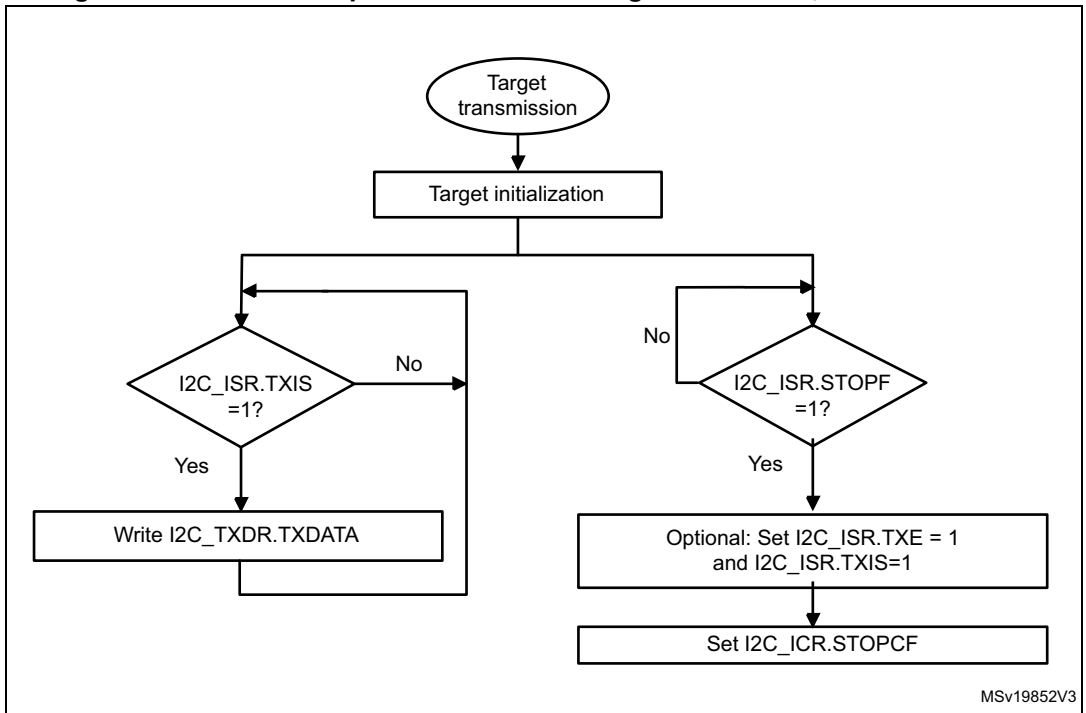
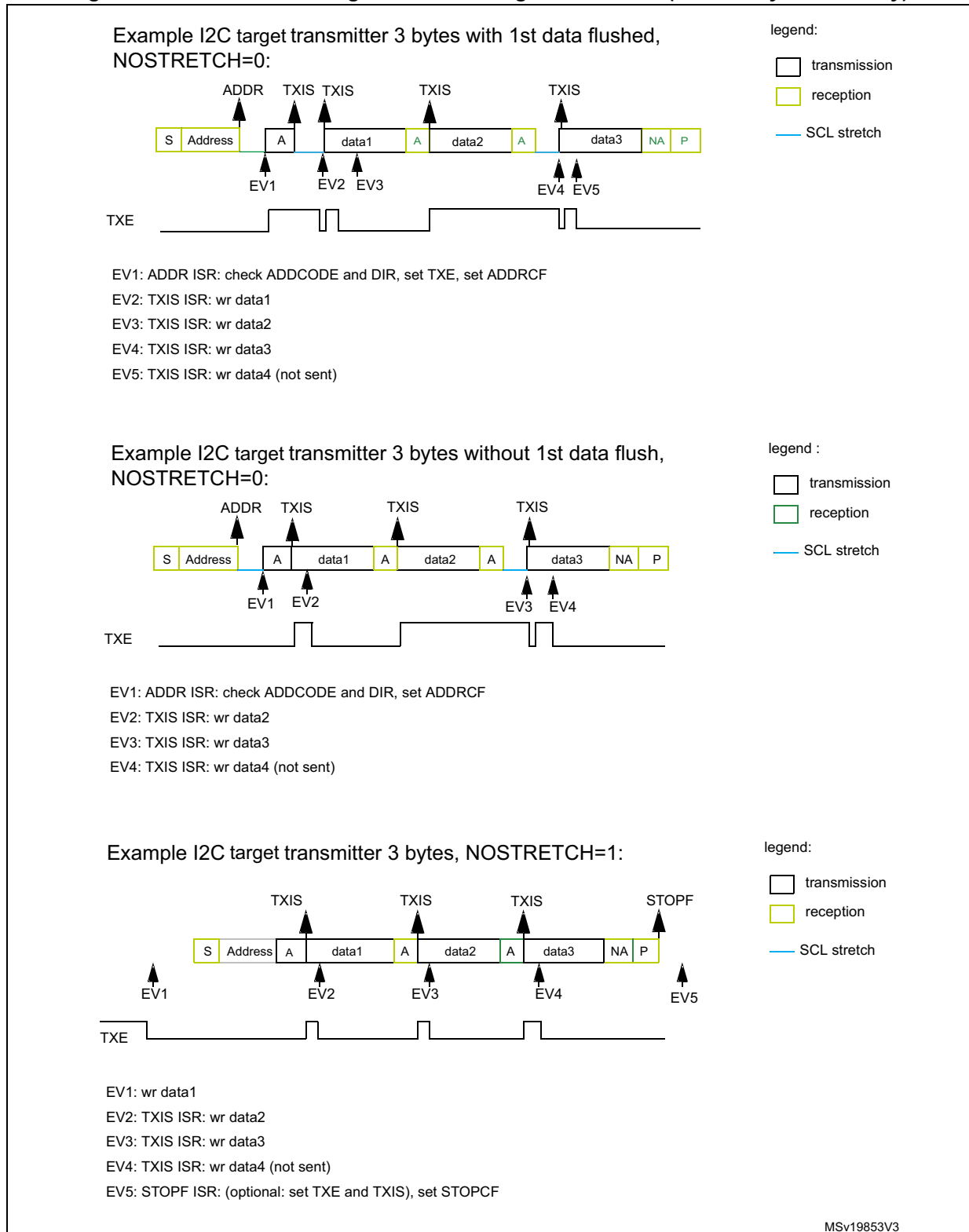


Figure 293. Transfer bus diagrams for I2C target transmitter (mandatory events only)



**Target receiver**

The RXNE bit of the I2C\_ISR register is set when the I2C\_RXDR is full, which generates an interrupt if the RXIE bit of the I2C\_CR1 register is set. RXNE is cleared when I2C\_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C\_CR1 register is set, the STOPF flag in the I2C\_ISR register is set and an interrupt is generated.

**Figure 294. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0**

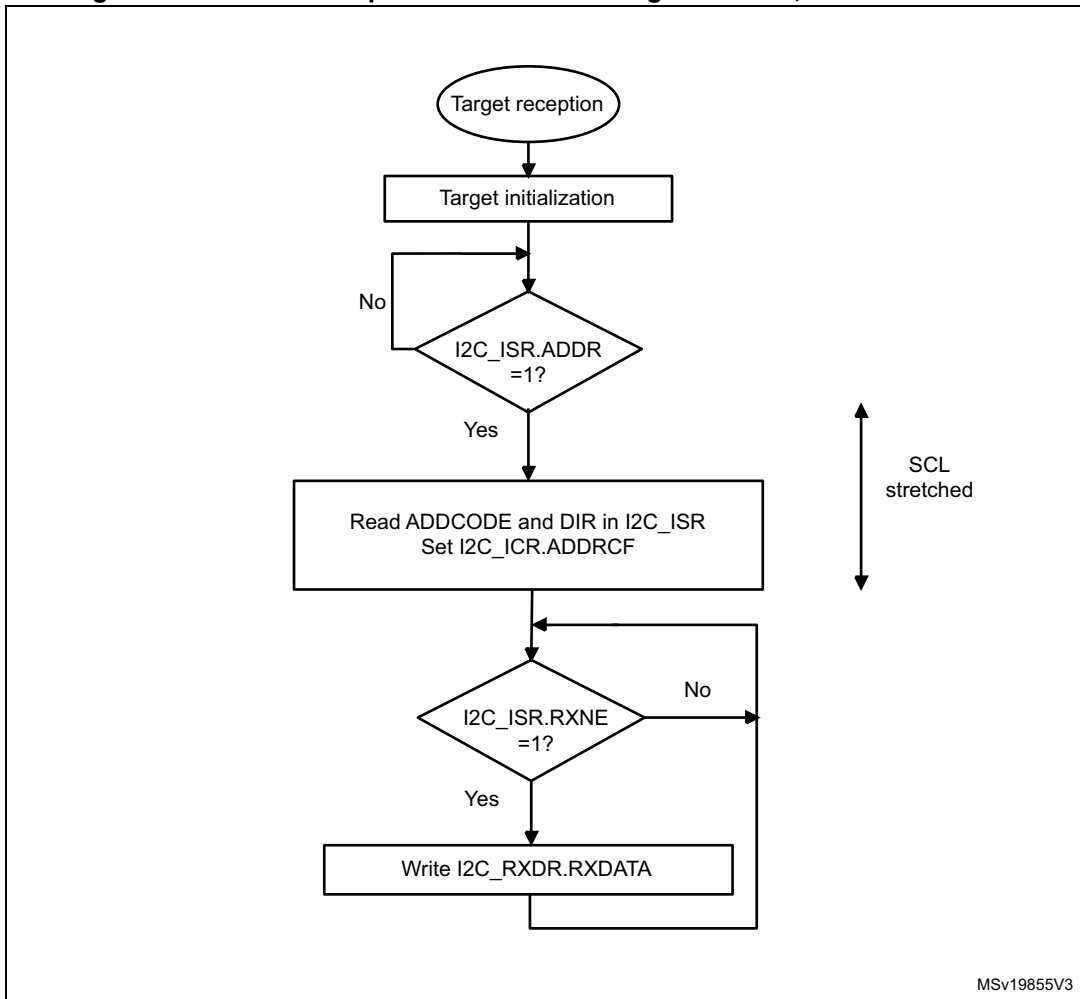


Figure 295. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

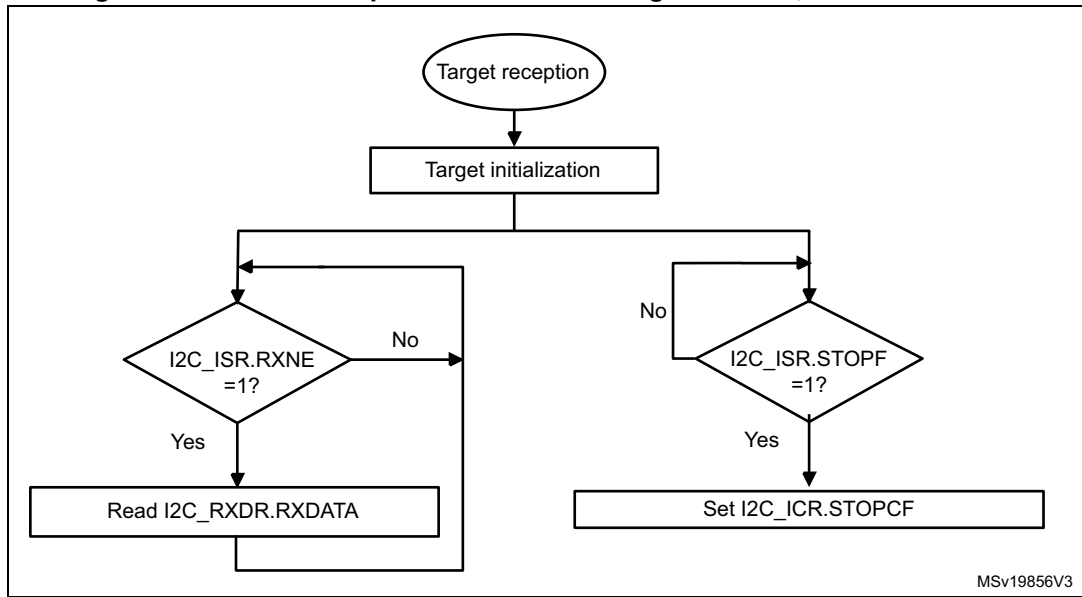
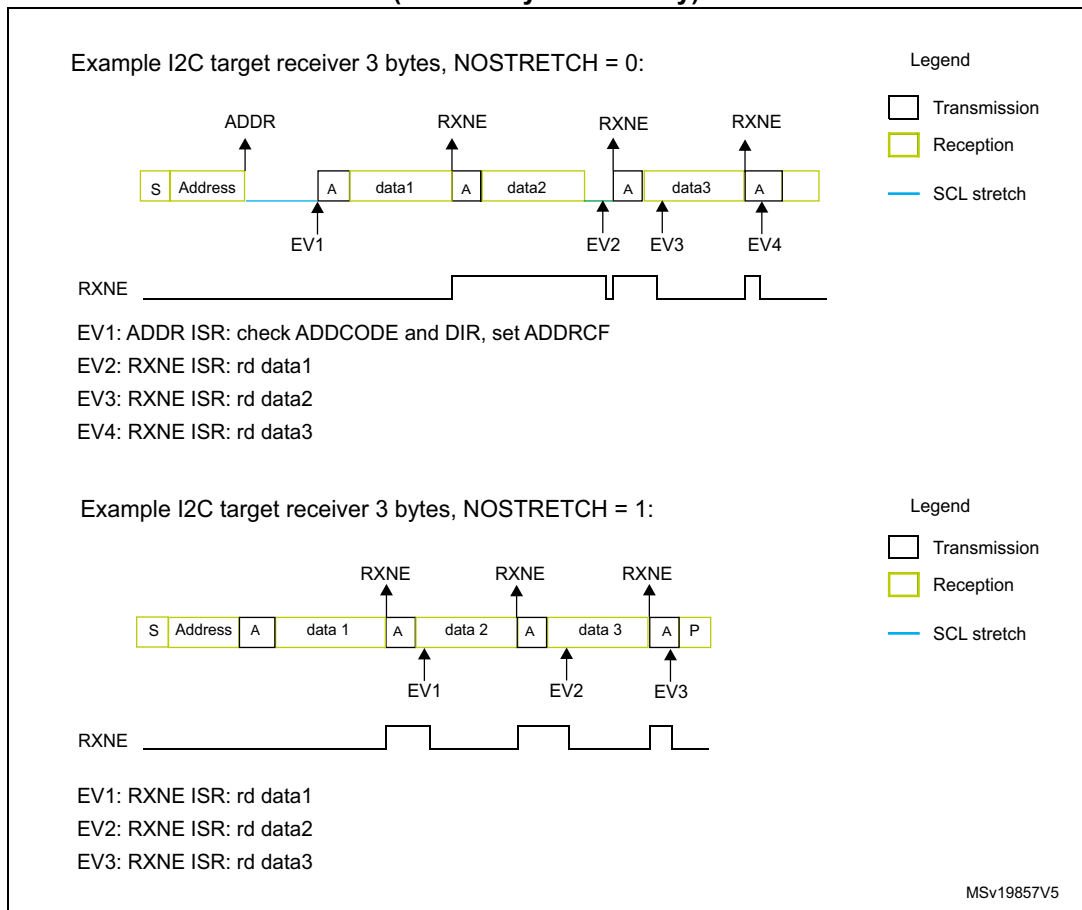


Figure 296. Transfer bus diagrams for I2C target receiver (mandatory events only)



### 32.4.9 I2C controller mode

#### I2C controller initialization

Before enabling the peripheral, the I2C controller 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 multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a  $t_{\text{SYNC1}}$  delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C\_TIMINGR register.

I2C detects its own SCL high level after a  $t_{\text{SYNC2}}$  delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C\_TIMINGR register.

Consequently the controller clock period is:

$$t_{\text{SCL}} = t_{\text{SYNC1}} + t_{\text{SYNC2}} + \{[(\text{SCLH} + 1) + (\text{SCLL} + 1)] \times (\text{PRESC} + 1) \times t_{\text{I2CCLK}}\}$$

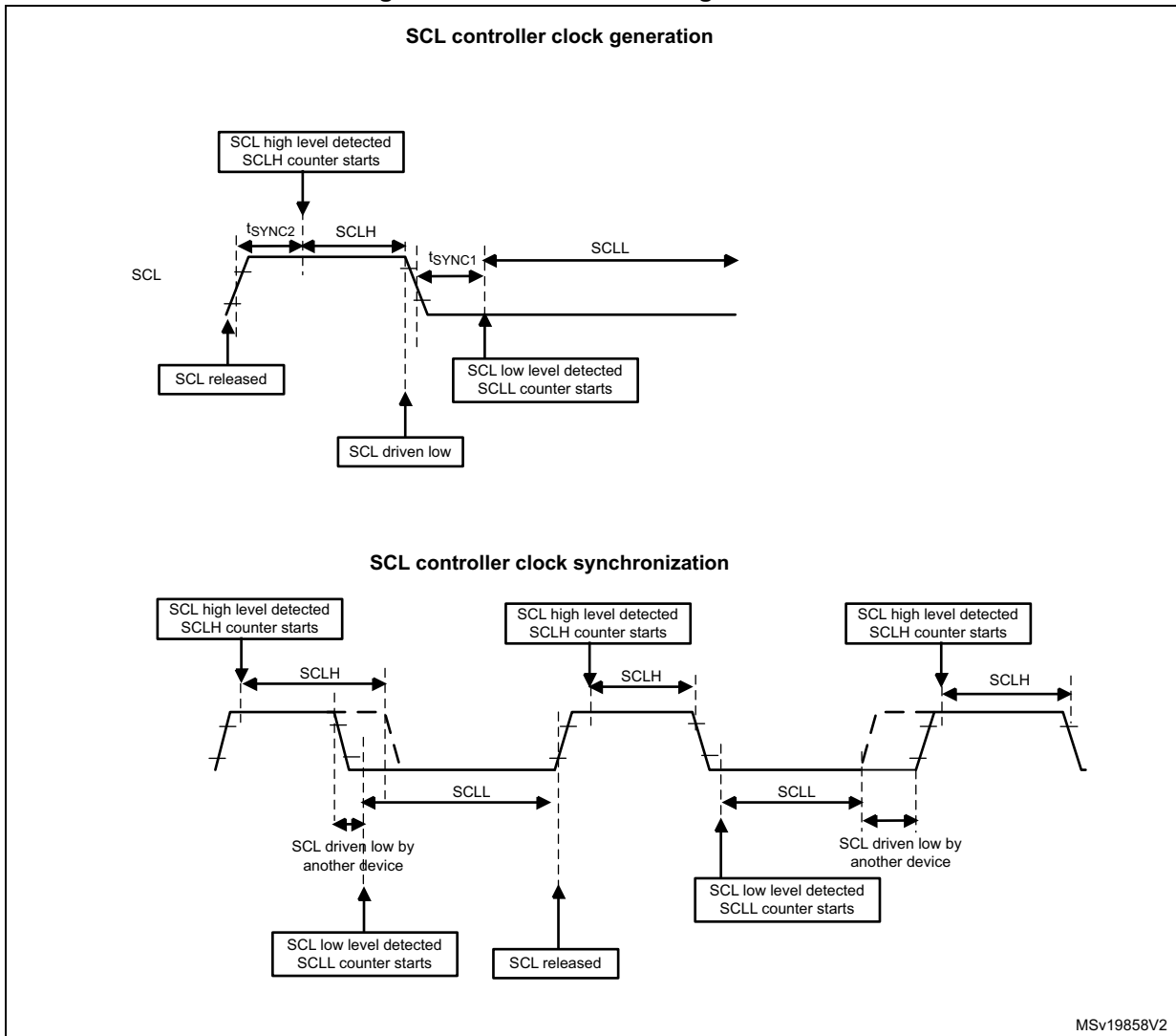
The duration of  $t_{\text{SYNC1}}$  depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled):  $\text{DNF}[3:0] \times t_{\text{I2CCLK}}$
- delay due to SCL synchronization with the I2CCLK clock (two to three I2CCLK periods)

The duration of  $t_{\text{SYNC2}}$  depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled):  $\text{DNF}[3:0] \times t_{\text{I2CCLK}}$
- delay due to SCL synchronization with the I2CCLK clock (two to three I2CCLK periods)

Figure 297. Controller clock generation





**Caution:** For compliance with the I<sup>2</sup>C-bus or SMBus specification, the controller clock must respect the timings in the following table.

**Table 171. I<sup>2</sup>C-bus and 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>	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t <sub>r</sub>	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t <sub>f</sub>	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

**Note:** The SCLL[7:0] bitfield also determines the t<sub>BUF</sub> and t<sub>SU:STA</sub> timings and SCLH[7:0] the t<sub>HD:STA</sub> and t<sub>SU:STO</sub> timings.

Refer to [Section 32.4.10](#) for examples of I2C\_TIMINGR settings versus the I2CCLK frequency.

### Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C\_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD\_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

**Note:** Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I<sup>2</sup>C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I<sup>2</sup>C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C\_CR1 register.

To launch the communication, set the START bit of the I2C\_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or  $t_{BUF}$  time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

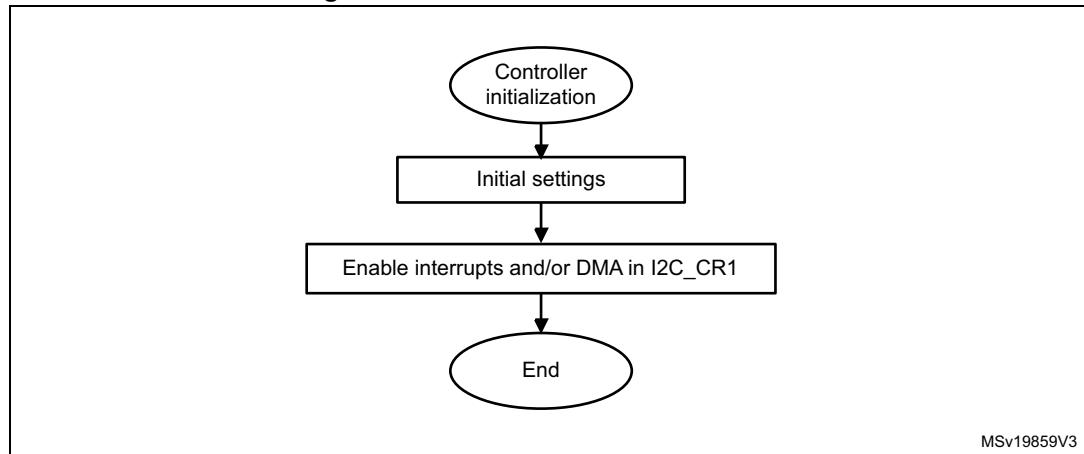
*Note:* The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

*In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCONF bit makes I2C quit that loop.*

*If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared when the ADDRCONF bit is set.*

*Note:* The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

**Figure 298. Controller initialization flow**



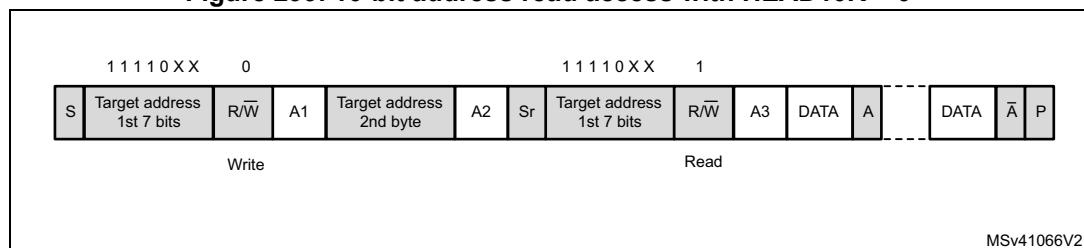
MSv19859V3

**Initialization of a controller receiver addressing a 10-bit address target**

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C\_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

**Figure 299. 10-bit address read access with HEAD10R = 0**

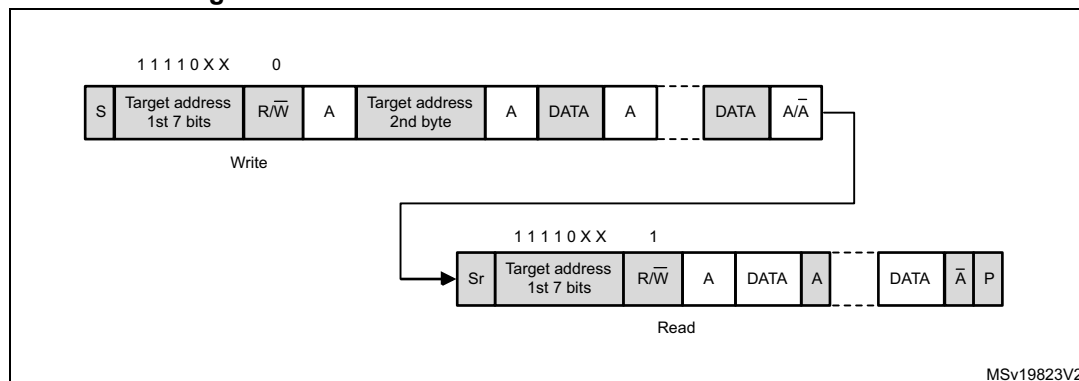


MSv41066V2

If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

**Figure 300. 10-bit address read access with HEAD10R = 1**



**Controller transmitter**

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C\_CR1 register is set. The flag is cleared when the I2C\_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C\_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C\_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C\_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C\_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 301. Transfer sequence flow for I2C controller transmitter,  $N \leq 255$  bytes

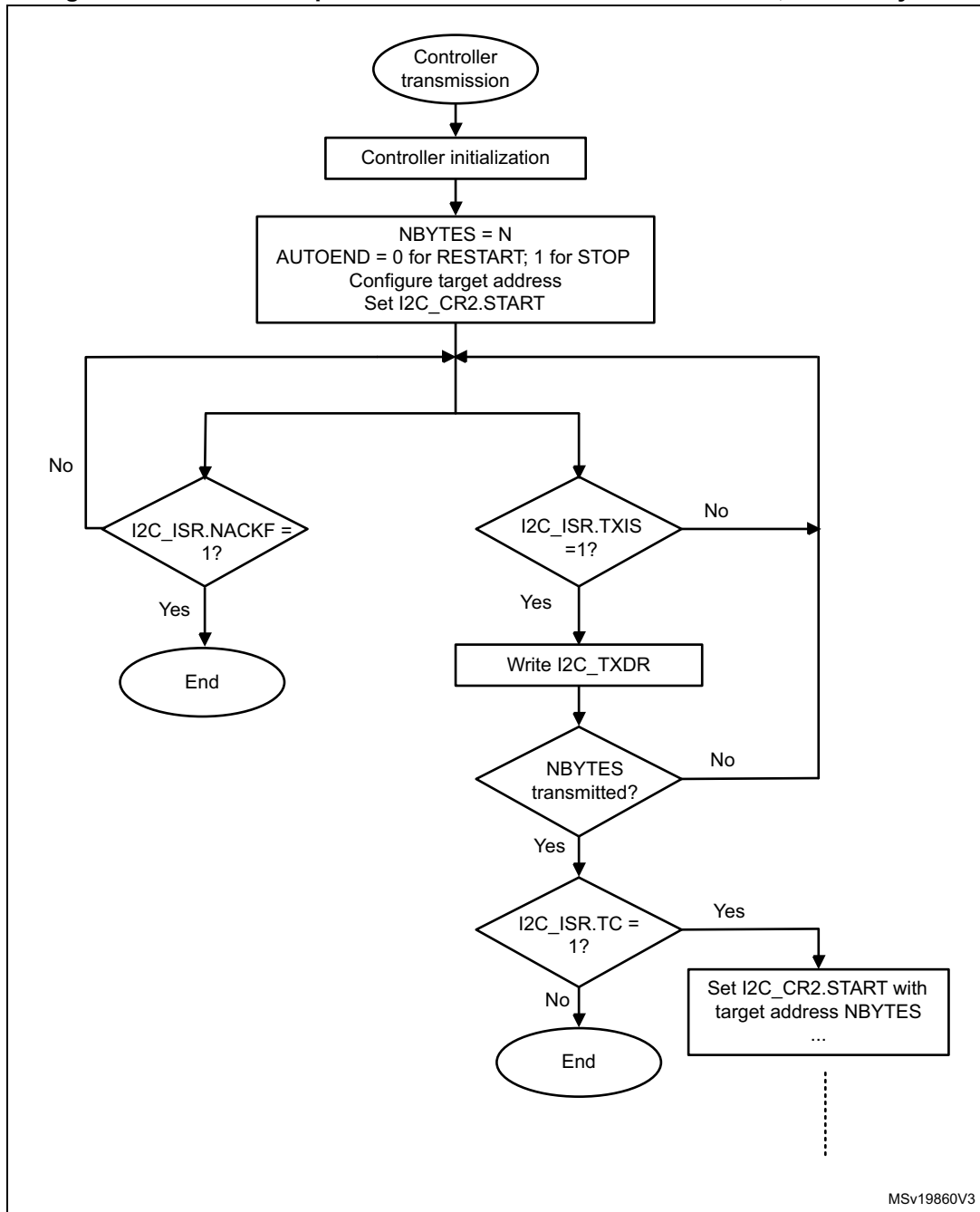
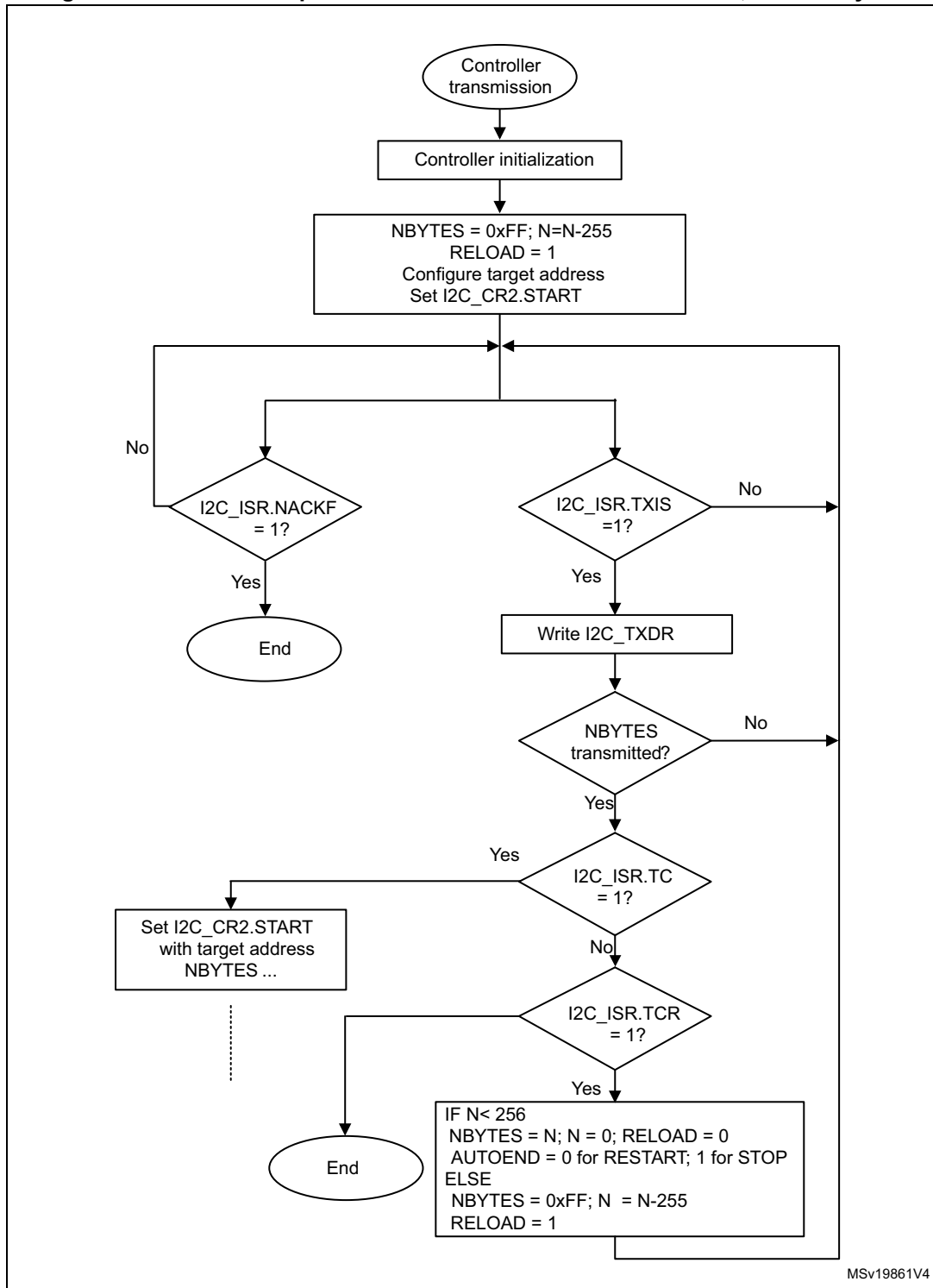
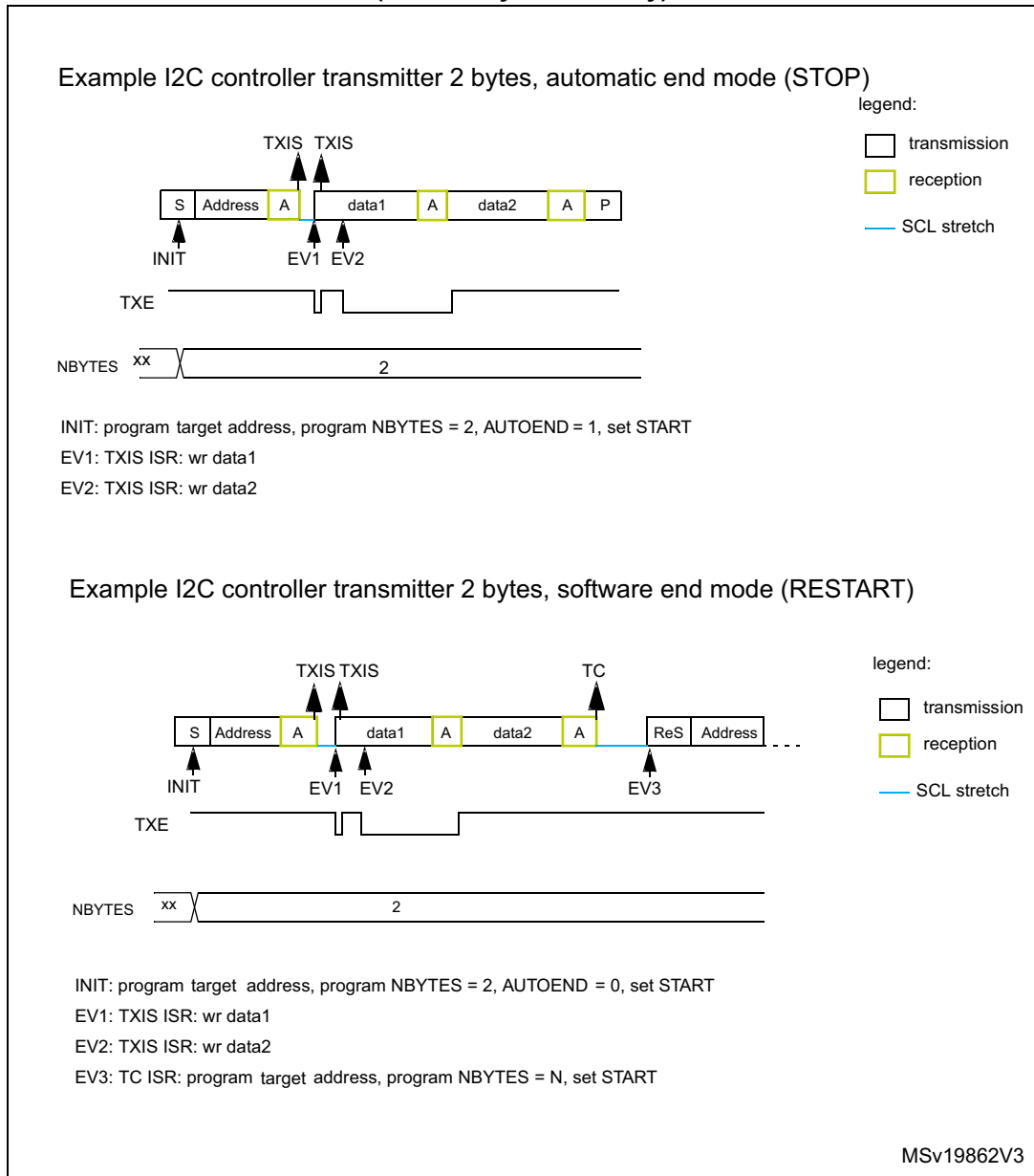


Figure 302. Transfer sequence flow for I2C controller transmitter, N > 255 bytes



**Figure 303. Transfer bus diagrams for I2C controller transmitter (mandatory events only)**



### Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C\_CR1 register is set. The flag is cleared when I2C\_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C\_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C\_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C\_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 304. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

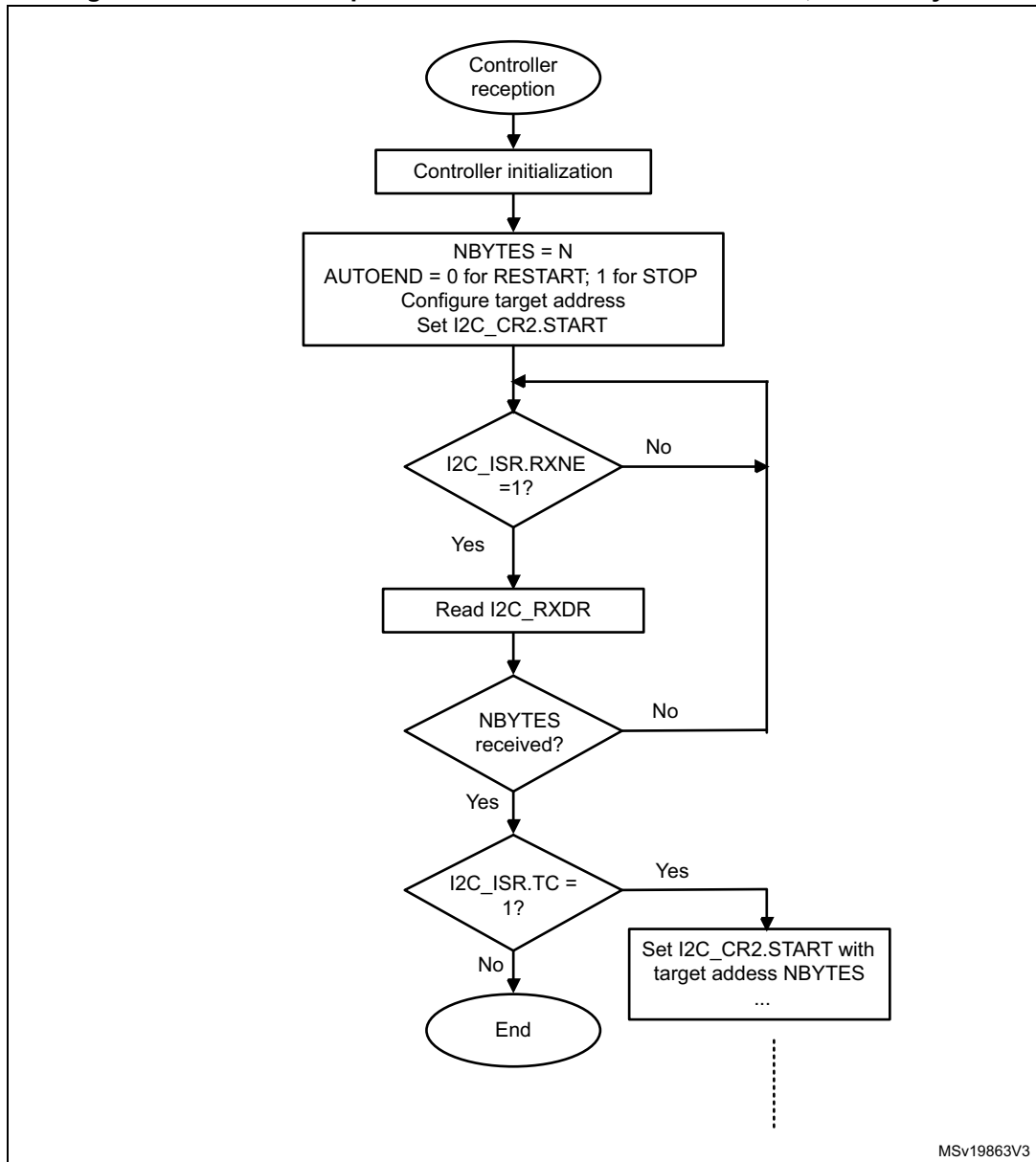
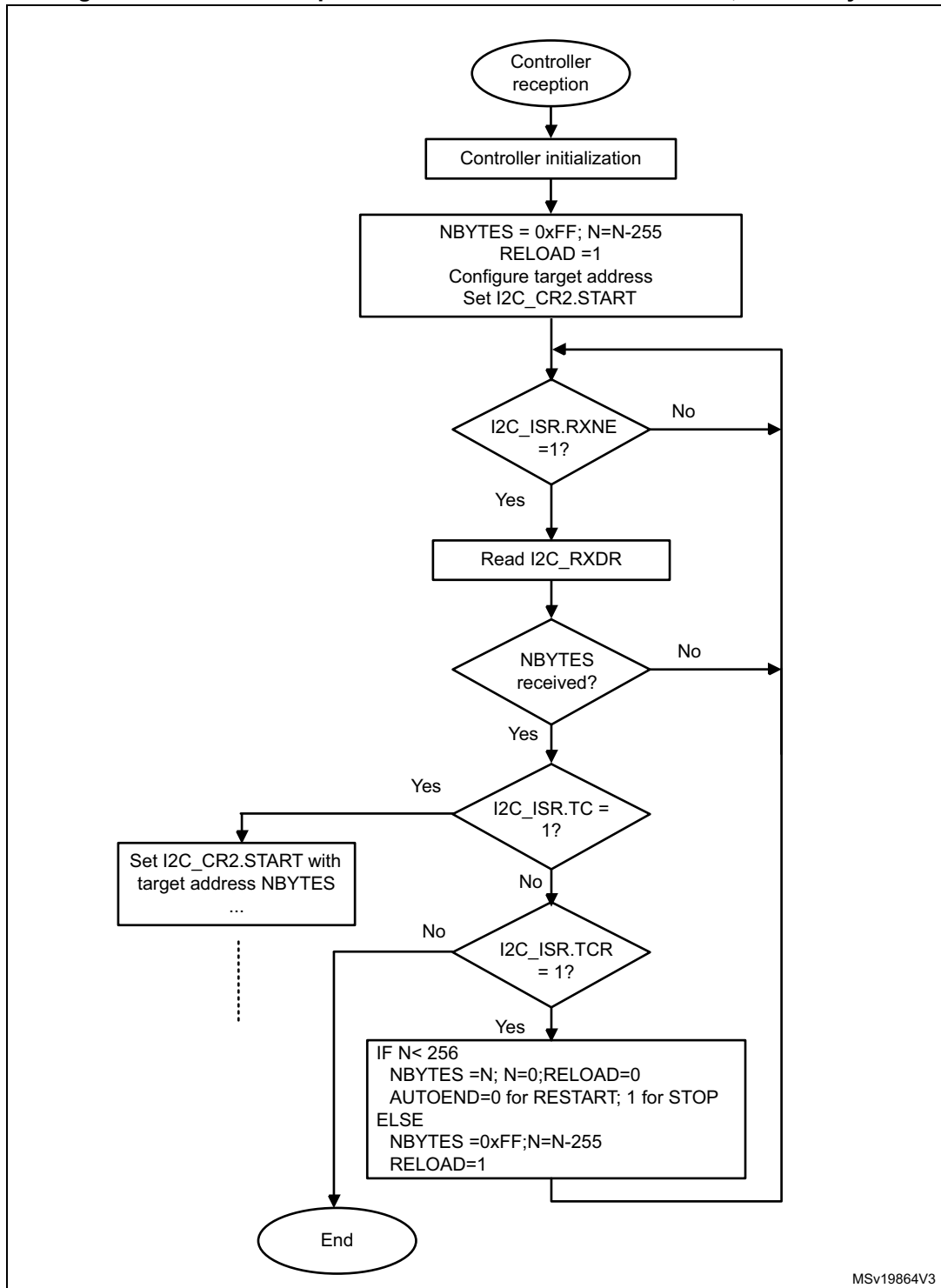
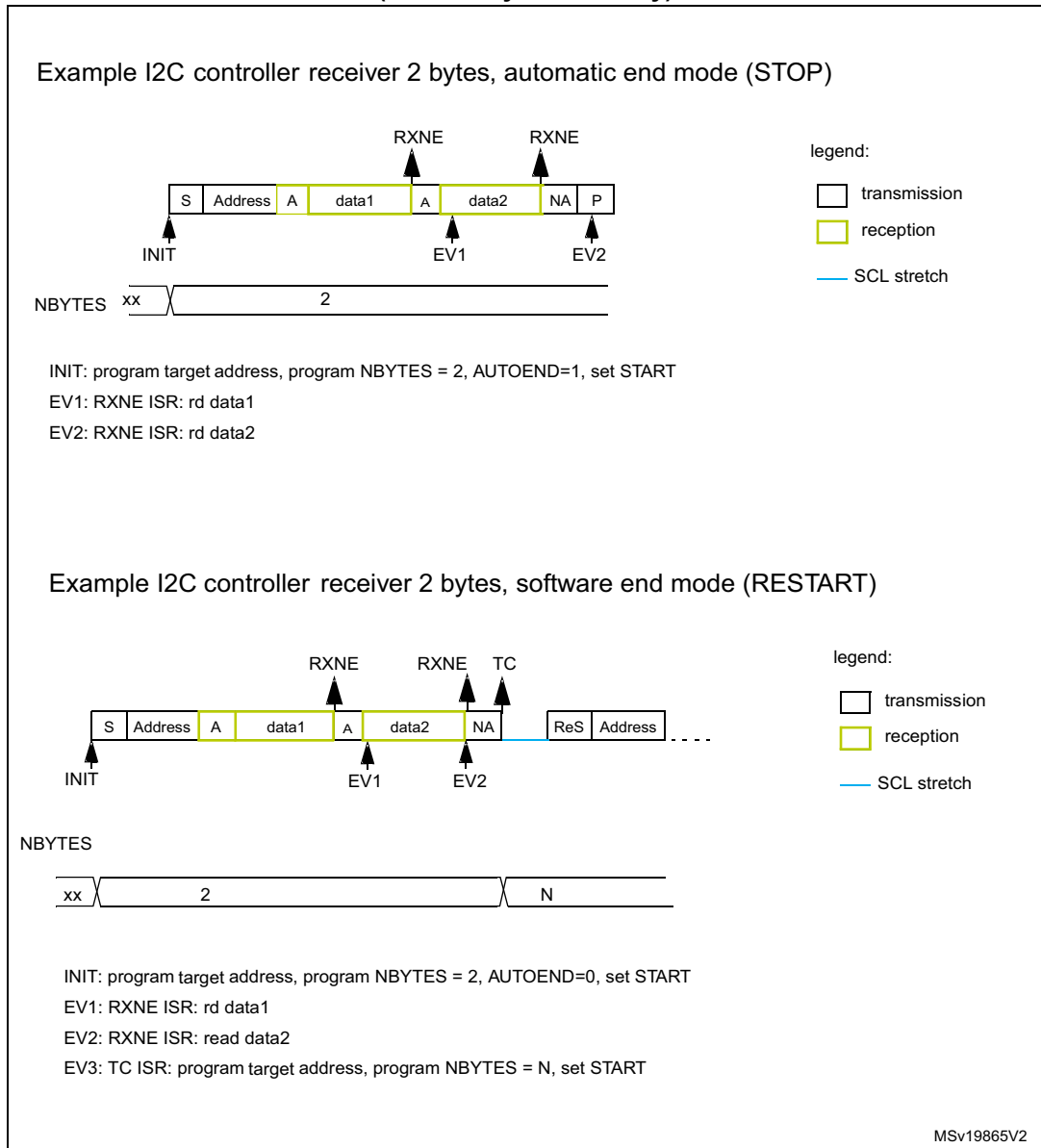




Figure 305. Transfer sequence flow for I2C controller receiver, N > 255 bytes



**Figure 306. Transfer bus diagrams for I2C controller receiver (mandatory events only)**



### 32.4.10 I2C\_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C\_TIMINGR register to obtain timings compliant with the I<sup>2</sup>C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 172. Timing settings for  $f_{I2CCLK}$  of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
$t_{SCLL}$	200 x 250 ns = 50 $\mu$ s	20 x 250 ns = 5.0 $\mu$ s	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
$t_{SCLH}$	196 x 250 ns = 49 $\mu$ s	16 x 250 ns = 4.0 $\mu$ s	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
$t_{SCL}^{(1)}$	~100 $\mu$ s <sup>(2)</sup>	~10 $\mu$ s <sup>(2)</sup>	~2.5 $\mu$ s <sup>(3)</sup>	~2.0 $\mu$ s <sup>(4)</sup>
SDADEL[3:0]	0x2	0x2	0x1	0x0
$t_{SDADEL}$	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

- $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 1000$  ns.
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 750$  ns.
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 500$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 655$  ns.

Table 173. Timing settings for  $f_{I2CCLK}$  of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
$t_{SCLL}$	200 x 250 ns = 50 $\mu$ s	20 x 250 ns = 5.0 $\mu$ s	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
$t_{SCLH}$	196 x 250 ns = 49 $\mu$ s	16 x 250 ns = 4.0 $\mu$ s	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
$t_{SCL}^{(1)}$	~100 $\mu$ s <sup>(2)</sup>	~10 $\mu$ s <sup>(2)</sup>	~2.5 $\mu$ s <sup>(3)</sup>	~1.0 $\mu$ s <sup>(4)</sup>
SDADEL[3:0]	0x2	0x2	0x2	0x0
$t_{SDADEL}$	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
$t_{SCLDEL}$	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

- $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to SCL internal detection delay. Values provided for  $t_{SCL}$  are examples only.
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 1000$  ns.
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 750$  ns.
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 250$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 500$  ns.

Table 174. Timing settings for  $f_{I2CCLK}$  of 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0xB	0xB	0x5	0x5
SCLL[7:0]	0xC7	0x13	0x9	0x3
$t_{SCLL}$	200 x 250 ns = 50 $\mu$ s	20 x 250 ns = 5.0 $\mu$ s	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH[7:0]	0xC3	0xF	0x3	0x1
$t_{SCLH}$	196 x 250 ns = 49 $\mu$ s	16 x 250 ns = 4.0 $\mu$ s	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
$t_{SCL}^{(1)}$	~100 $\mu$ s <sup>(2)</sup>	~10 $\mu$ s <sup>(2)</sup>	~2.5 $\mu$ s <sup>(3)</sup>	~875 ns <sup>(4)</sup>
SDADEL[3:0]	0x2	0x2	0x3	0x0
$t_{SDADEL}$	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
$t_{SCLDEL}$	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

- $t_{SCL}$  is greater than  $t_{SCLL} + t_{SCLH}$  due to the SCL internal detection delay. Values provided for  $t_{SCL}$  are only examples.
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 1000$  ns
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 750$  ns
- $t_{SYNC1} + t_{SYNC2}$  minimum value is  $4 \times t_{I2CCLK} = 83.3$  ns. Example with  $t_{SYNC1} + t_{SYNC2} = 250$  ns

### 32.4.11 SMBus specific features

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

#### Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I<sup>2</sup>C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (<http://smbus.org>).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

#### Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write*

*Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write, and Block Write-Block Read Process Call.* The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (<http://smbus.org>).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance supporting SMBus.

### Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C\_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (<http://smbus.org>).

### Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C\_CR1 register. Refer to [Target byte control mode](#) for more details.

### Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C\_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

### SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C\_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C\_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

*Note:* If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

**Packet error checking**

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the  $C(x) = x^8 + x^2 + x + 1$  CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

**Timeouts**

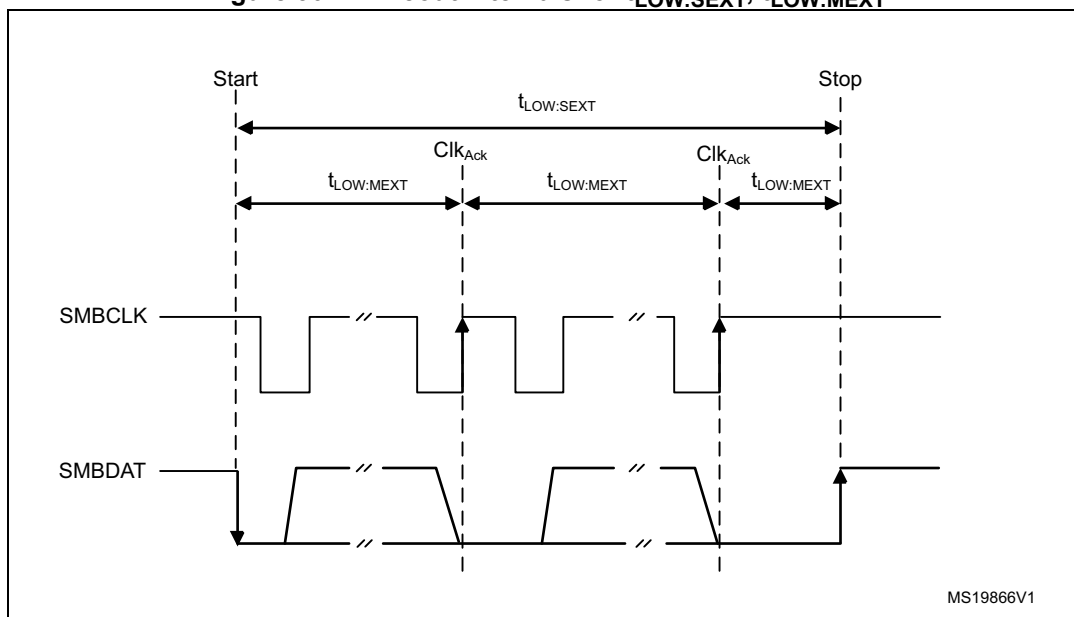
To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

**Table 175. SMBus timeout specifications**

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{\text{TIMEOUT}}$	Detect clock low timeout	25	35	ms
$t_{\text{LOW:SEXT}}^{(1)}$	Cumulative clock low extend time (target device)	-	25	
$t_{\text{LOW:MEXT}}^{(2)}$	Cumulative clock low extend time (controller device)	-	10	

- $t_{\text{LOW:SEXT}}$  is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than  $t_{\text{LOW:SEXT}}$ . The value provided applies to a single target device connected to a full-target controller.
- $t_{\text{LOW:MEXT}}$  is the cumulative time a controller 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 target device or another controller also extends the clock, causing the combined clock low time to be greater than  $t_{\text{LOW:MEXT}}$  on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 307. Timeout intervals for  $t_{LOW:SEXT}$ ,  $t_{LOW:MEXT}$



MS19866V1

### Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for  $t_{IDLE} > t_{HIGH(max)}$  (refer to the table in [Section 32.4.9](#)).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

### 32.4.12 SMBus initialization

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

In addition to the I2C initialization for the I<sup>2</sup>C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

#### Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C\_CR1 register. Refer to [Target byte control mode](#) for more details.

#### Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to [Bus idle detection](#) for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C\_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C\_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C\_CR1 register.

### Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C\_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C\_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

**Caution:** Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

**Table 176. SMBus with PEC configuration**

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

### Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C\_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

#### *t<sub>TIMEOUT</sub> check*

To check the  $t_{\text{TIMEOUT}}$  parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C\_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the  $(\text{TIMEOUTA} + 1) \times 2048 \times t_{\text{I2CCLK}}$  period, the TIMEOUT flag of the I2C\_ISR register is set.

Refer to [Table 177](#).

**Caution:** Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMOUTEN bit is set.

#### *t<sub>LOW:SEXT</sub> and t<sub>LOW:MEXT</sub> check*

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking  $t_{\text{LOW:SEXT}}$  for the I2C peripheral operating as a target, or  $t_{\text{LOW:MEXT}}$  when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C\_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the  $(\text{TIMEOUTB} + 1) \times 2048 \times t_{\text{I2CCLK}}$  period, and within the timeout interval described in [Bus idle detection](#) section, the TIMEOUT flag of the I2C\_ISR register is set.

Refer to [Table 178](#).



**Caution:** Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

### Bus idle detection

To check the  $t_{IDLE}$  period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMEOUTEN bit of the I2C\_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the  $(TIMEOUTA + 1) \times 4 \times t_{I2CCLK}$  period, the TIMEOUT flag of the I2C\_ISR register is set.

Refer to [Table 179](#).

**Caution:** Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

### 32.4.13 SMBus I2C\_TIMEOUTR register configuration examples

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

The following tables provide examples of settings to reach desired  $t_{TIMEOUT}$ ,  $t_{LOW:SEXT}$ ,  $t_{LOW:MEXT}$ , and  $t_{IDLE}$  timings at different  $f_{I2CCLK}$  frequencies.

**Table 177. TIMEOUTA[11:0] for maximum  $t_{TIMEOUT}$  of 25 ms**

$f_{I2CCLK}$	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	$98 \times 2048 \times 125 \text{ ns} = 25 \text{ ms}$
16 MHz	0xC3	0	1	$196 \times 2048 \times 62.5 \text{ ns} = 25 \text{ ms}$
48 MHz	0x249	0	1	$586 \times 2048 \times 20.08 \text{ ns} = 25 \text{ ms}$

**Table 178. TIMEOUTB[11:0] for maximum  $t_{LOW:SEXT}$  and  $t_{LOW:MEXT}$  of 8 ms**

$f_{I2CCLK}$	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	$32 \times 2048 \times 125 \text{ ns} = 8 \text{ ms}$
16 MHz	0x3F	1	$64 \times 2048 \times 62.5 \text{ ns} = 8 \text{ ms}$
48 MHz	0xBB	1	$188 \times 2048 \times 20.08 \text{ ns} = 8 \text{ ms}$

**Table 179. TIMEOUTA[11:0] for maximum  $t_{IDLE}$  of 50  $\mu\text{s}$**

$f_{I2CCLK}$	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{IDLE}$
8 MHz	0x63	1	1	$100 \times 4 \times 125 \text{ ns} = 50 \mu\text{s}$
16 MHz	0xC7	1	1	$200 \times 4 \times 62.5 \text{ ns} = 50 \mu\text{s}$
48 MHz	0x257	1	1	$600 \times 4 \times 20.08 \text{ ns} = 50 \mu\text{s}$

### 32.4.14 SMBus target mode

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

In addition to I2C target transfer management (refer to [Section 32.4.8: I2C target mode](#)), this section provides extra software flowcharts to support SMBus.

#### SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C\_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 308. Transfer sequence flow for SMBus target transmitter N bytes + PEC**

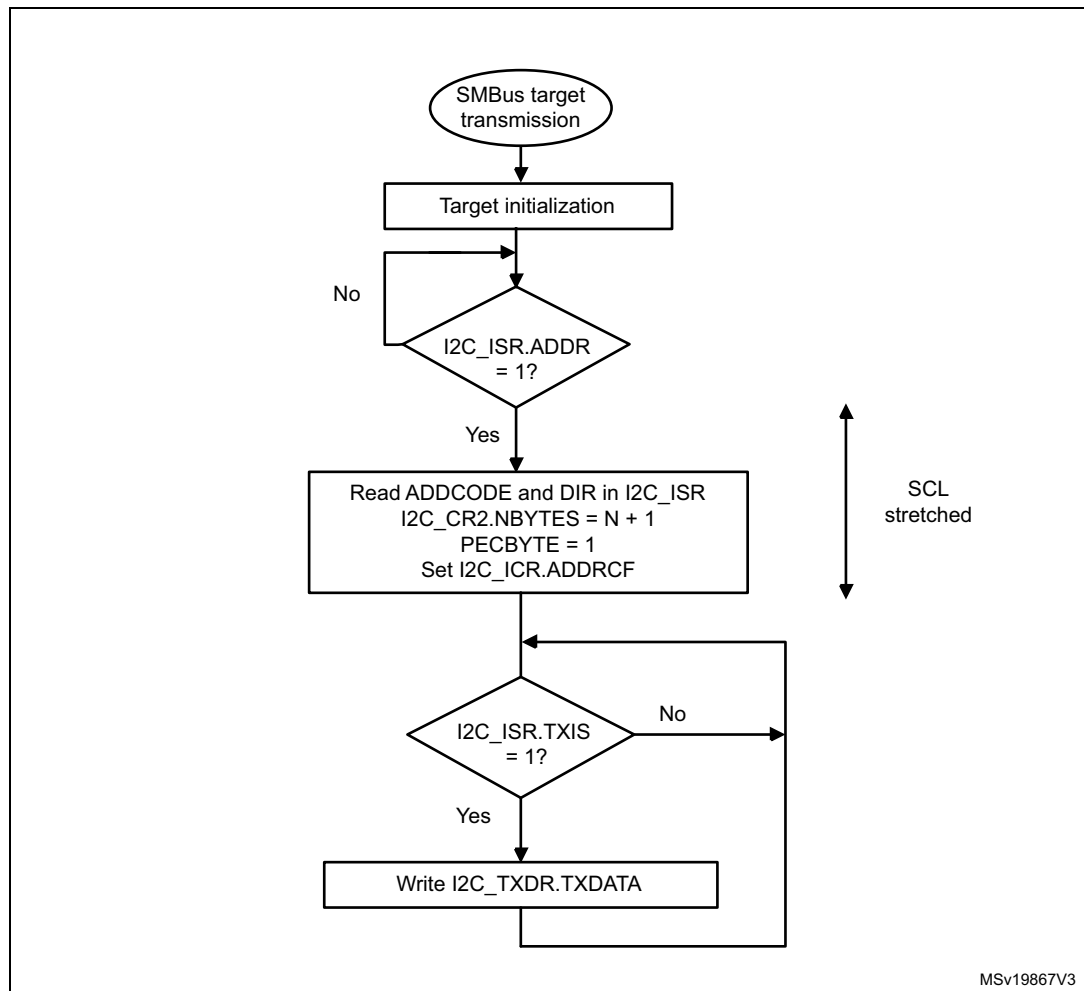
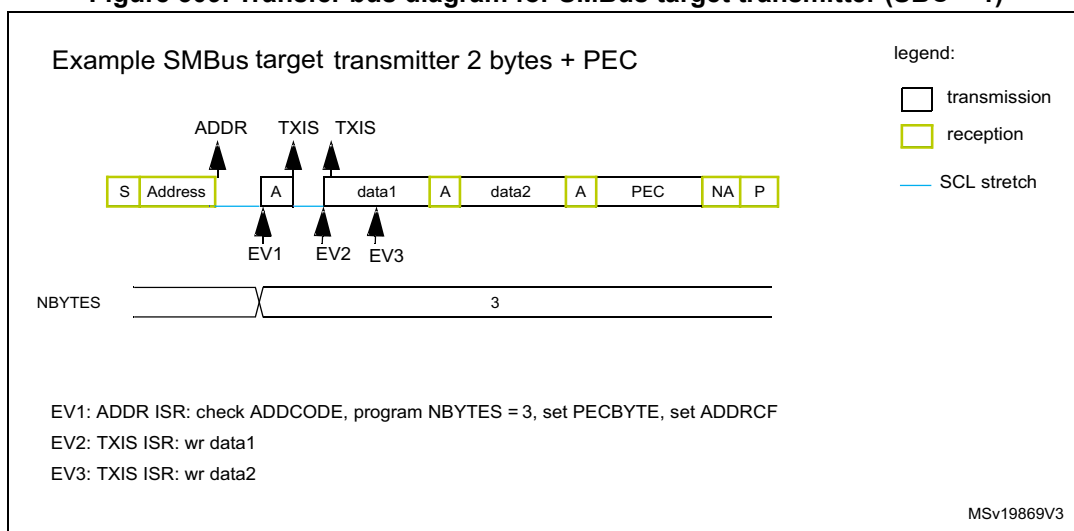


Figure 309. Transfer bus diagram for SMBus target transmitter (SBC = 1)



### SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C\_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C\_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 310. Transfer sequence flow for SMBus target receiver N bytes + PEC

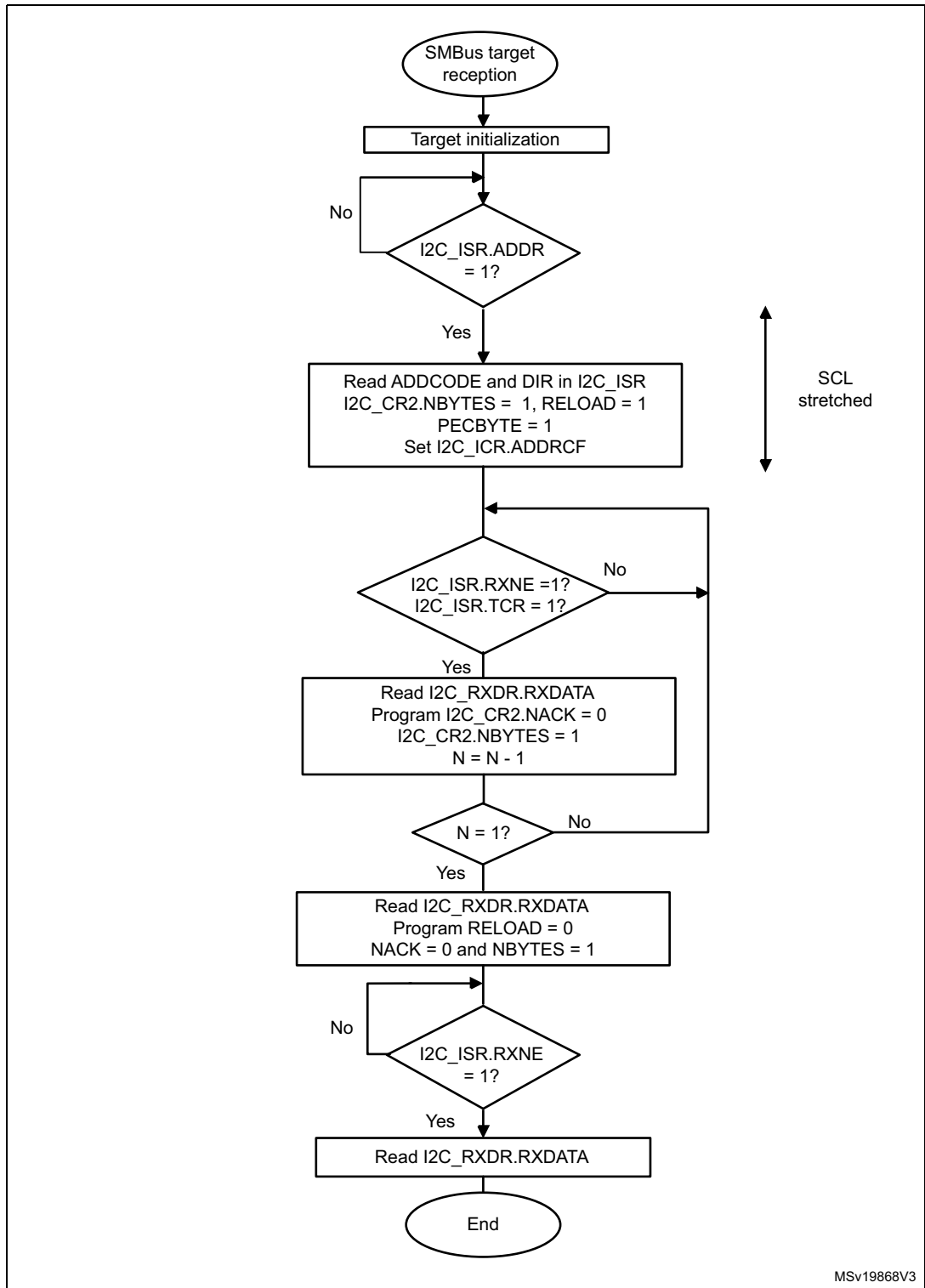
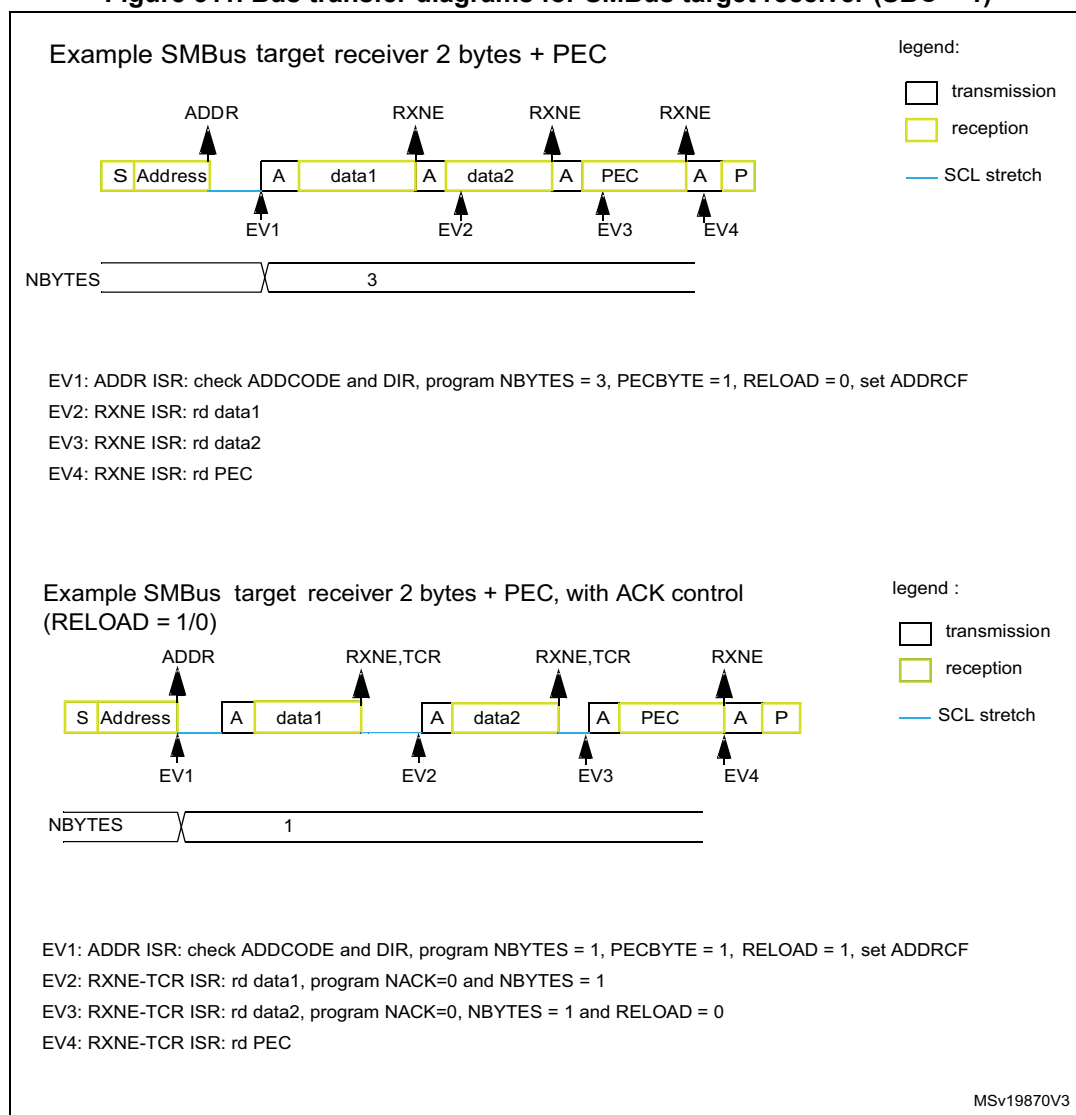


Figure 311. Bus transfer diagrams for SMBus target receiver (SBC = 1)



### 32.4.15 SMBus controller mode

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

In addition to I2C controller transfer management (refer to [Section 32.4.9: I2C controller mode](#)), this section provides extra software flowcharts to support SMBus.

#### SMBus controller transmitter

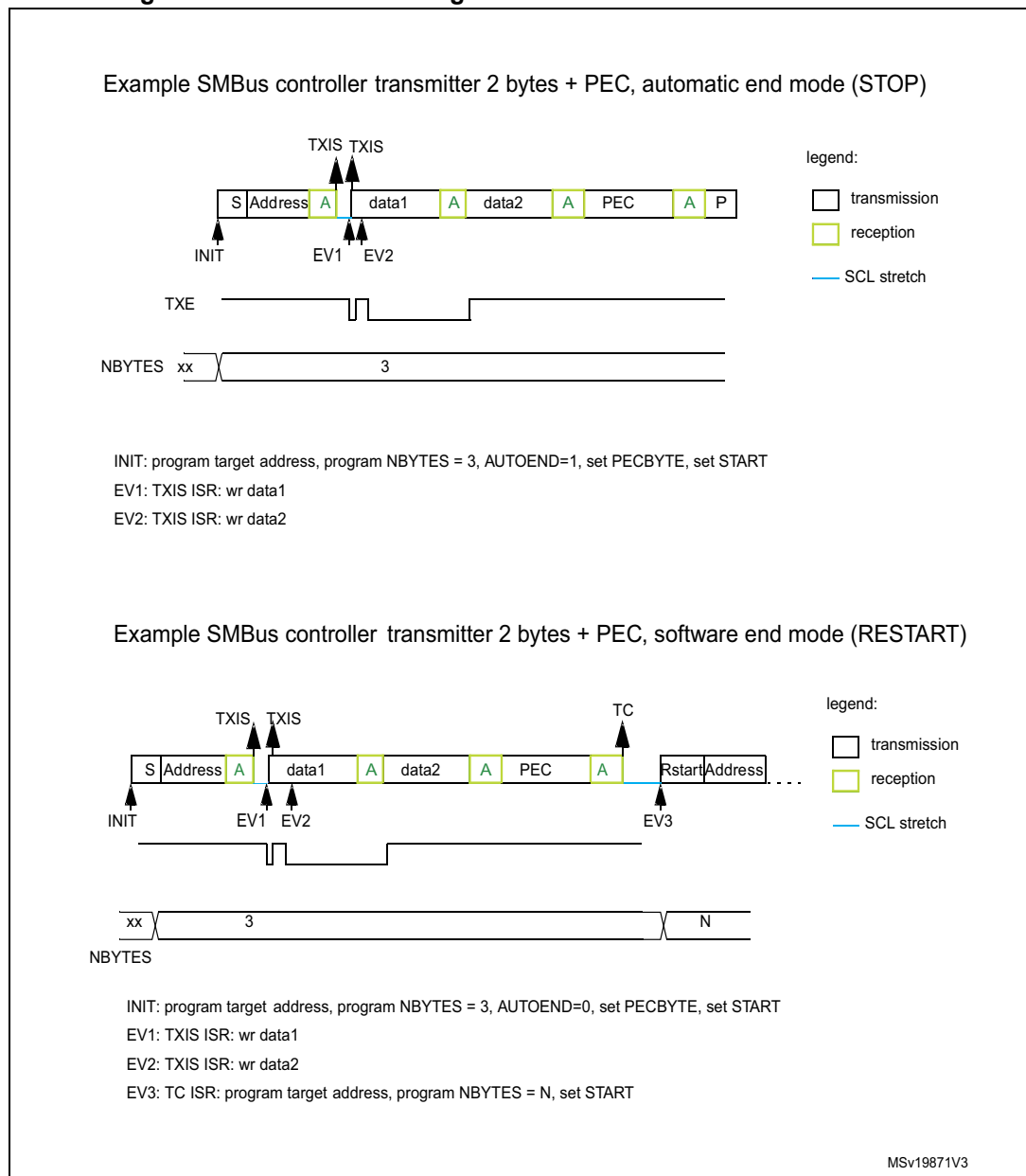
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C\_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C\_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 312. Bus transfer diagrams for SMBus controller transmitter**



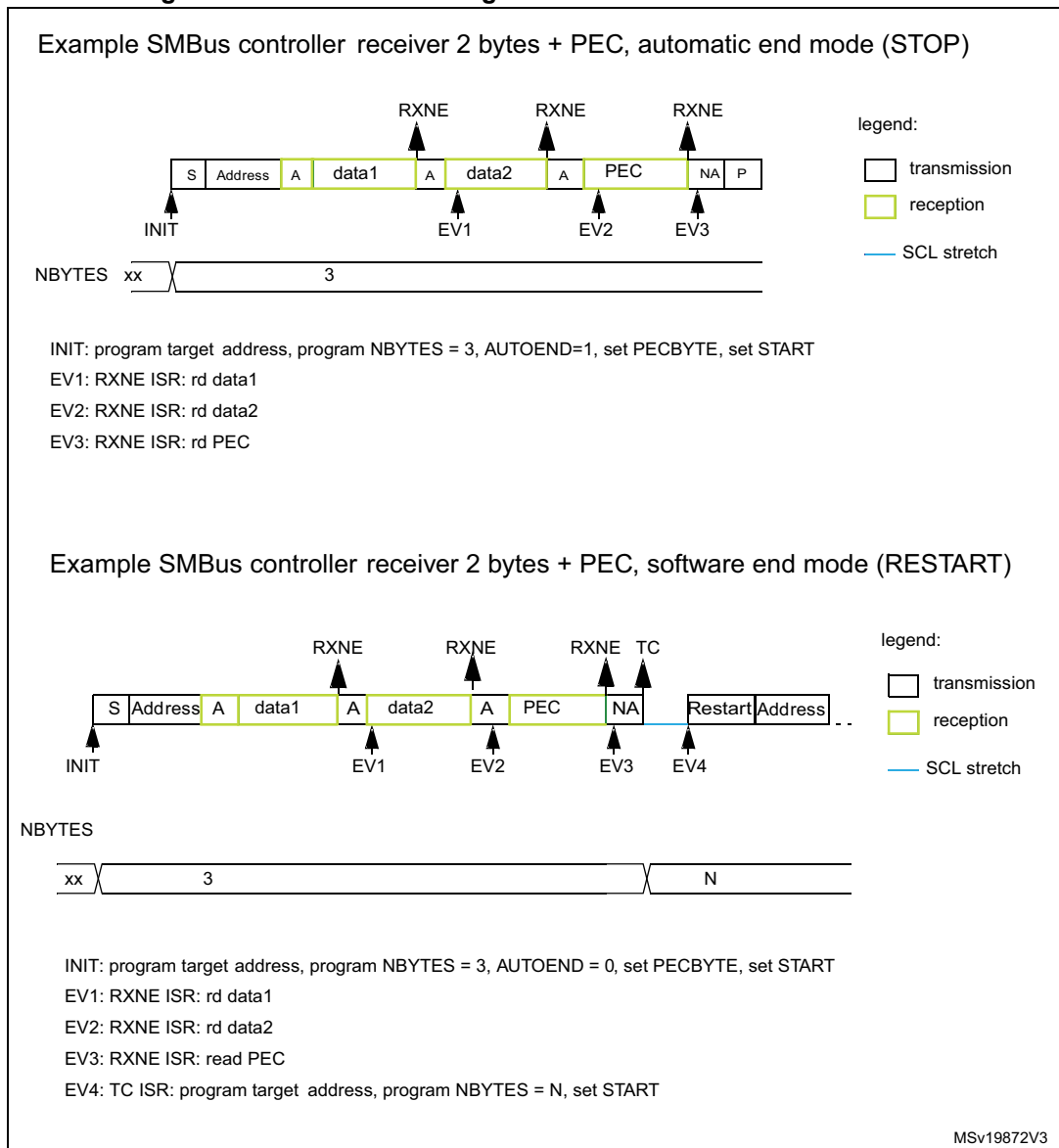
**SMBus controller receiver**

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C\_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C\_PECR register content. 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 313. Bus transfer diagrams for SMBus controller receiver



MSv19872V3

### 32.4.16 Wake-up from Stop mode on address match

This section pertains to I2C instances supporting the wake-up from Stop mode feature (refer to [Section 32.3](#)).

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C\_CR1 register. The HSI16 oscillator must be selected as the clock source for I2CCLK to allow the wake-up from Stop mode.

In Stop mode, the HSI16 oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI16 oscillator and stretches SCL low until the oscillator wakes up.

HSI16 is then used for the address reception.



If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI16 oscillator is stopped again and the device does not wake up.

**Note:** *When the system clock is used as I2C clock, or when WUPEN = 0, the HSI16 oscillator does not start upon receiving START condition.*

*Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.*

**Caution:** The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

**Caution:** The feature is only available when the HSI16 oscillator is selected as the I2C clock.

**Caution:** Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

**Caution:** If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

### 32.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

#### Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C\_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

#### Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller 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 controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C\_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

### Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C\_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C\_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C\_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

### Packet error checking error (PECERR)

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

A PEC error is detected when the received PEC byte does not match the I2C\_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C\_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

### Timeout error (TIMEOUT)

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus  $t_{\text{LOW:MEXT}}$  parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus  $t_{\text{LOW:SEXT}}$  parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C\_ISR register and an interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

### Alert (ALERT)

This section pertains to the instances of the I2C peripheral supporting SMBus. Refer to [Section 32.3: I2C implementation](#).

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C\_CR1 register is set.

### 32.5 I2C in low-power modes

Table 180. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop <sup>(1)</sup>	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI16). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: the I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to [Section 32.3: I2C implementation](#) for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

### 32.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 181. I2C interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby modes
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 ADDRCF = 1			
NACK reception	NACKF	NACKIE	Write NACKCF = 1	No			
I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
	Arbitration loss	ARLO		Write ARLOCF = 1			
	Overrun/underrun	OVR		Write OVRCF = 1			

Table 181. I2C interrupt requests (continued)

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby modes
I2C_ERR	PEC error	PECERR	ERRIE	Write PECERRCF = 1	Yes	No	No
	Timeout/ $t_{LOW}$ error	TIMEOUT		Write TIMEOUTCF = 1			
	SMBus alert	ALERT		Write ALERTCF = 1			

- The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to [Section 32.3: I2C implementation](#).

## 32.7 I2C DMA requests

### 32.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C\_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see [Section 10: Direct memory access controller \(DMA\)](#)) to the I2C\_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to [Controller transmitter](#).

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to [SMBus target transmitter](#) and [SMBus controller transmitter](#). This applies to the instances of the I2C peripheral supporting SMBus.

*Note:* If DMA is used for transmission, it is not required to set the TXIE bit.

### 32.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C\_CR1 register. Data is loaded from the I2C\_RXDR register to an SRAM area configured through the DMA peripheral (refer to [Section 10: Direct memory access controller \(DMA\)](#)) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*. This applies to the instances of the I2C peripheral supporting SMBus.

*Note:* If DMA is used for reception, it is not required to set the RXIE bit.

### 32.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG\_I2Cx\_SMBUS\_TIMEOUT bits in the DBG block.

### 32.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

#### 32.9.1 I2C control register 1 (I2C\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x PCLK + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERT EN	SMBDEN	SMBHEN	GCEN	WUPE N	NOSTRETCH	SBC
								r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMA AEN	TXDMA EN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOP IE	NACKIE	ADDRI E	RXIE	TXIE	PE
r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 23 **PECEN**: PEC enable

- 0: PEC calculation disabled
- 1: PEC calculation enabled

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to *Section 32.3*.

- Bit 22 **ALERTEN**: SMBus alert enable
- 0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
  - 1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).
- On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).
- Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.*
- Bit 21 **SMBDEN**: SMBus device default address enable
- 0: Device default address disabled. Address 0b1100001x is NACKed.
  - 1: Device default address enabled. Address 0b1100001x is ACKed.
- On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).
- Bit 20 **SMBHEN**: SMBus host address enable
- 0: Host address disabled. Address 0b0001000x is NACKed.
  - 1: Host address enabled. Address 0b0001000x is ACKed.
- On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).
- 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 disabled.
  - 1: Wake-up from Stop mode enabled.
- On the instances of the I2C peripheral that do not support the wake-up from Stop mode feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).
- Note: WUPEN can be set only when DNF[3:0] = 0000.*
- Bit 17 **NOSTRETCH**: Clock stretching disable
- This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
- 0: Clock stretching enabled
  - 1: Clock stretching disabled
- Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).*
- Bit 16 **SBC**: Target byte control
- This bit is used to enable hardware byte control in target mode.
- 0: Target byte control disabled
  - 1: Target byte control enabled
- Bit 15 **RXDMAEN**: DMA reception requests enable
- 0: DMA mode disabled for reception
  - 1: DMA mode enabled for reception
- Bit 14 **TXDMAEN**: DMA transmission requests enable
- 0: DMA mode disabled for transmission
  - 1: DMA mode enabled for transmission
- Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF**: Analog noise filter OFF

0: Analog noise filter enabled

1: Analog noise filter disabled

*Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).*

Bits 11:8 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to  $DNF[3:0] * t_{I2CCLK}$

0000: Digital filter disabled

0001: Digital filter enabled and filtering capability up to one  $t_{I2CCLK}$

...

1111: digital filter enabled and filtering capability up to fifteen  $t_{I2CCLK}$

*Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).*

Bit 7 **ERRIE**: Error interrupts enable

0: Error detection interrupts disabled

1: Error detection interrupts enabled

*Note: Any of these errors generates an interrupt:*

- arbitration loss (ARLO)

- bus error detection (BERR)

- overrun/underrun (OVR)

- timeout detection (TIMEOUT)

- PEC error detection (PECERR)

- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable

0: Transfer complete interrupt disabled

1: Transfer complete interrupt enabled

*Note: Any of these events generates an interrupt:*

Transfer complete (TC)

Transfer complete reload (TCR)

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 (target 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 disabled  
 1: Peripheral enabled

*Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.*

### 32.9.2 I2C control register 2 (I2C\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	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	HEAD1OR	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

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).

*Note: Writing 0 to this bit has no effect.*

*This bit has no effect when RELOAD is set, and in target mode when SBC = 0.*

Bit 25 **AUTOEND**: Automatic end mode (controller 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 target 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 target mode with SBC = 0.

*Note: Changing these bits when the START bit is set is not allowed.*



**Bit 15 NACK:** NACK generation (target 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 only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.*

*When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.*

*When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.*

**Bit 14 STOP:** STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

- 0: No STOP generation
- 1: STOP generation after current byte transfer

*Note: Writing 0 to this bit has no effect.*

**Bit 13 START:** START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software, by setting the ADDRCONF bit of the I2C\_ICR register.

- 0: No START generation
- 1: RESTART/START generation:

If the I2C is already in controller 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 target mode.*

*This bit has no effect when RELOAD is set.*

**Bit 12 HEAD10R:** 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

*Note: Changing this bit when the START bit is set is not allowed.*

**Bit 11 ADD10:** 10-bit addressing mode (controller mode)

- 0: The controller operates in 7-bit addressing mode
- 1: The controller 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 (controller mode)

- 0: Controller requests a write transfer
- 1: Controller requests a read transfer

*Note: Changing this bit when the START bit is set is not allowed.*

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

**Condition: In 7-bit addressing mode (ADD10 = 0):**

SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

**Condition: In 10-bit addressing mode (ADD10 = 1):**

SADD[9:0] must be written with the 10-bit target address to be sent.

*Note: Changing these bits when the START bit is set is not allowed.*

### 32.9.3 I2C own address 1 register (I2C\_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x PCLK + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1M ODE	OA1[9:0]									
r/w					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.

1: Own address 1 enabled. The received target 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 target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

*Note: These bits can be written only when OA1EN = 0.*

### 32.9.4 I2C own address 2 register (I2C\_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x PCLK + 6 x I2CCLK.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	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 target address OA2 is NACKed.

1: Own address 2 enabled. The received target 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 ≠ 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.

### 32.9.5 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]			
r/w	r/w	r/w	r/w					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale I2CCLK to generate the clock period  $t_{PRESC}$  used for data setup and hold counters (refer to section [I2C timings](#)), and for SCL high and low level counters (refer to section [I2C controller initialization](#)).

$$t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$$

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay  $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$  between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during  $t_{SCLDEL}$ .

*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 controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during  $t_{SDADEL}$ .

$$t_{SDADEL} = SDADEL \times t_{PRESC}$$

*Note:  $SDADEL$  is used to generate  $t_{HD:DAT}$  timing.*

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

$$t_{SCLH} = (SCLH + 1) \times t_{PRESC}$$

*Note:  $SCLH$  is also used to generate  $t_{SU:STO}$  and  $t_{HD:STA}$  timing.*

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

$$t_{SCLL} = (SCLL + 1) \times t_{PRESC}$$

*Note:  $SCLL$  is also used to generate  $t_{BUF}$  and  $t_{SU:STA}$  timings.*

*Note: This register must be configured when the I2C peripheral is disabled (PE = 0).*

*Note: The STM32CubeMX tool calculates and provides the I2C\_TIMINGR content in the I2C Configuration window.*

### 32.9.6 I2C timeout register (I2C\_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to  $2 \times PCLK + 6 \times I2CCLK$ .

On the instances of the I2C peripheral that do not support the SMBus feature, this register is reserved, and its bits are forced by hardware to 0. Refer to [Section 32.3](#).

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

**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:  
 – Controller mode: the controller cumulative clock low extend time ( $t_{LOW:MEXT}$ ) is detected  
 – Target mode: the target cumulative clock low extend time ( $t_{LOW:SEXT}$ ) is detected  
 $t_{LOW:EXT} = (TIMEOUTB + TIDLE = 01) \times 2048 \times t_{I2CCLK}$   
*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  
 $t_{TIMEOUT} = (TIMEOUTA + 1) \times 2048 \times t_{I2CCLK}$   
 The bus idle condition (both SCL and SDA high) when TIDLE = 1  
 $t_{IDLE} = (TIMEOUTA + 1) \times 4 \times t_{I2CCLK}$   
*Note: These bits can be written only when TIMOUTEN = 0.*

### 32.9.7 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]						DIR	
								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	TIMEOUT	PECERR	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 (target 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 (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).

*Note: This bit is cleared by hardware when PE = 0.*

Bit 12 **TIMEOUT**: Timeout or t<sub>LOW</sub> detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).

*Note: This bit is cleared by hardware when PE = 0.*

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).

*Note: This bit is cleared by hardware when PE = 0.*

- Bit 10 **OVR**: Overrun/underrun (target mode)  
This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRDCF 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 target mode. It is cleared by software by setting the BERRCF bit.  
*Note: This bit is cleared by hardware when PE = 0.*
- Bit 7 **TCR**: Transfer complete reload  
This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.  
*Note: This bit is cleared by hardware when PE = 0.*  
*This flag is only for controller mode, or for target mode when the SBC bit is set.*
- Bit 6 **TC**: Transfer complete (controller 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:
- as a controller, provided that the STOP condition is generated by the peripheral.
  - as a target, 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 (target mode)  
This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting ADDRDCF 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 only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

*Note: This bit is cleared by hardware when PE = 0.*

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C\_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C\_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C\_TXDR.

*Note: This bit is set by hardware when PE = 0.*

### 32.9.8 I2C interrupt clear register (I2C\_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

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.

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C\_ISR register.

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).

Bit 11 **PECCF**: PEC error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C\_ISR register.

On the instances of the I2C peripheral that do not support the SMBus feature, this bit is reserved and forced by hardware to 0. Refer to [Section 32.3](#).

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.

### 32.9.9 I2C PEC register (I2C\_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

On the instances of the I2C peripheral that do not support the SMBus feature, this register is reserved and its bits are forced by hardware to 0. Refer to [Section 32.3](#).

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.	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.

### 32.9.10 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.	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.	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.

**32.9.11 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.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **TXDATA[7:0]**: 8-bit transmit dataData byte to be transmitted to the I<sup>2</sup>C-bus*Note: These bits can be written only when TXE = 1.*

### 32.9.12 I2C register map

The table below provides the I2C register map and the reset values.

**Table 182. 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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0x00	I2C_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]			ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE	
	Reset value										0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]											
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0						0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	Res.	OA2MSK[2:0]	OA2[7:1]					Res.				
	Reset value																	0						0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]			Res.	Res.	Res.	SCLDEL[3:0]	SDADEL[3:0]	SCLH[7:0]					SCLL[7:0]																			
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	Res.	TIMEOUTB[11:0]							TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]																
	Reset value	0																0					0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDRF	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																				0	0	0	0	0	0			0	0	0			
0x20	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.	PEC[7:0]						
	Reset value																											0	0	0	0	0	0	0
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.	RXDATA[7:0]						
	Reset value																											0	0	0	0	0	0	0
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.	TXDATA[7:0]						
	Reset value																											0	0	0	0	0	0	0

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



## 33 Universal synchronous receiver transmitter (USART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

### 33.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

## 33.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data  
Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

### 33.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

### 33.4 USART implementation

The table(s) below describe(s) USART implementation. It (they) also include(s) LPUART for comparison.

**Table 183. Instance implementation on STM32G0x1**

USART / LPUART instances	STM32G031xx, STM32G041xx STM32G051xx STM32G061xx	STM32G071xx, STM32G081xx	STM32G0B1xx, STM32G0C1xx
USART1	FULL	FULL	FULL
USART2	BASIC	FULL	FULL
USART3	-	BASIC	FULL
USART4	-	BASIC	BASIC
USART5	-	-	BASIC
USART6	-	-	BASIC
LPUART1	LP	LP	LP
LPUART2	-	-	LP

Table 184. USART / LPUART features

USART / LPUART modes/features <sup>(1)</sup>	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	X	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	-	-
LIN mode	X	-	-
Dual clock domain and wake-up from low-power mode	X	-	X
Receiver timeout interrupt	X	-	-
Modbus communication	X	-	-
Auto baud rate detection	X	-	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	-	X
Tx/Rx FIFO size	8	-	8
Prescaler	X	-	X
Wake-up from low-power mode	X <sup>(2)</sup>	-	X <sup>(2)</sup>

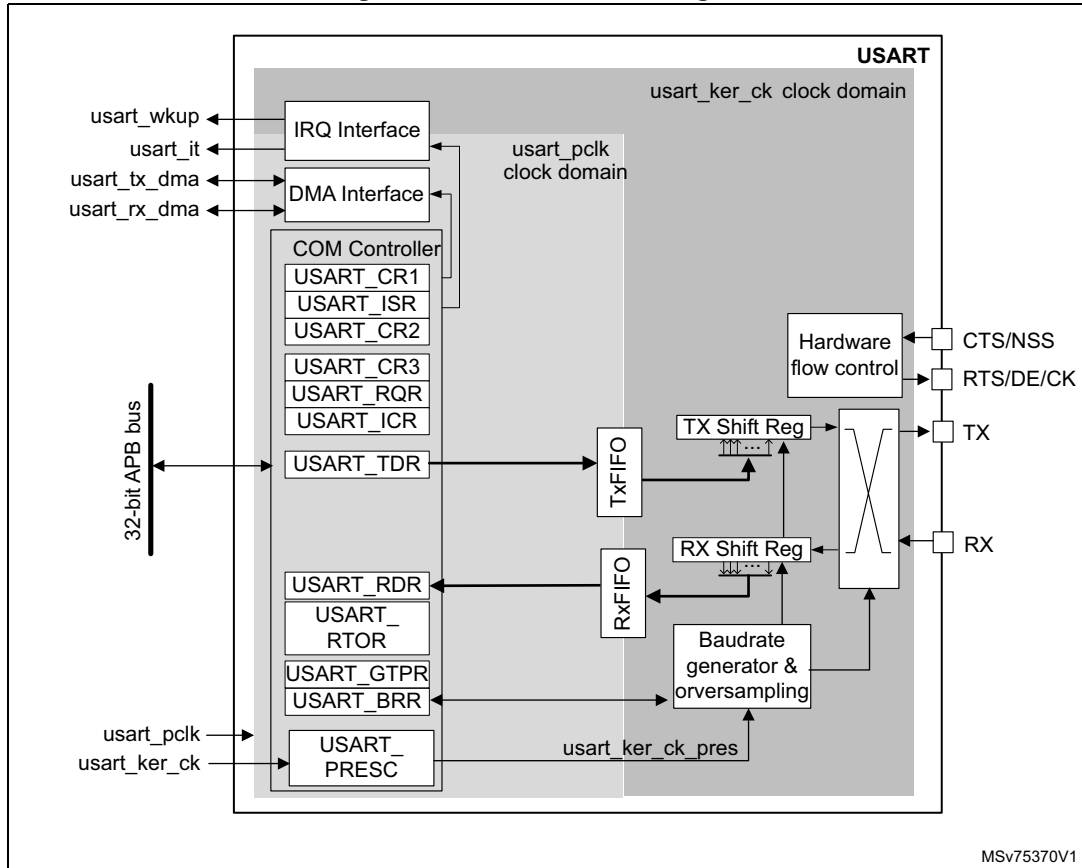
1. X = supported.

2. Wake-up supported from Stop 0 and Stop 1 modes.

### 33.5 USART functional description

#### 33.5.1 USART block diagram

Figure 314. USART block diagram



The simplified block diagram given in *Figure 314* shows two fully-independent clock domains:

- The **usart\_pclk** clock domain  
The **usart\_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart\_ker\_ck** kernel clock domain.  
The **usart\_ker\_ck** is the USART clock source. It is independent from **usart\_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart\_ker\_ck** clock is stopped.  
When the dual clock domain feature is disabled, the **usart\_ker\_ck** clock is the same as the **usart\_pclk** clock.

There is no constraint between **usart\_pclk** and **usart\_ker\_ck**: **usart\_ker\_ck** can be faster or slower than **usart\_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart\_ker\_ck** clock must be at least 3 times faster than the clock on the CK input.



## 33.5.2 USART signals

### USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)  
RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.
- **TX** (Transmit Data Output)  
When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

### RS232 hardware flow control mode

The following pins are required in RS232 hardware flow control mode:

- **CTS** (Clear To Send)  
When driven high, this signal blocks the data transmission at the end of the current transfer.
- **RTS** (Request To Send)  
When it is low, this signal indicates that the USART is ready to receive data.

### RS485 hardware flow control mode

The following pin is required in RS485 hardware control mode:

- **DE** (Driver Enable)  
This signal activates the transmission mode of the external transceiver.

### Synchronous SPI 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 SPI master and smartcard modes. It acts as clock input in synchronous SPI slave mode.  
In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.  
In smartcard mode, CK output provides the clock to the smartcard.
- **NSS**  
This pin acts as slave Select input in synchronous slave mode.

Refer to [Table 185](#) and [Table 186](#) for the list of USART input/output pins and internal signals.

**Table 185. USART input/output pins**

Pin name	Signal type	Description
USART_RX	Input	Serial data receive input
USART_TX	Output	Transmit data output
USART_CTS <sup>(2)</sup>	Input	Clear to send
USART_RTS <sup>(1)</sup>	Output	Request to send
USART_DE <sup>(1)</sup>	Output	Driver enable
USART_CK <sup>(1)</sup>	Input/Output	Clock input in synchronous slave mode. Clock output in synchronous master and smartcard modes.
USART_NSS <sup>(2)</sup>	Input	Slave select input in synchronous slave mode.

1. USART\_DE/USART\_RTS/USART\_CK share the same pin.
2. USART\_CTS and USART\_NSS share the same pin.

**Description of USART input/output signals**

**Table 186. USART internal input/output signals**

Pin name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request

**33.5.3 USART character description**

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART\_CR1 register (see [Figure 315](#)):

- 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.

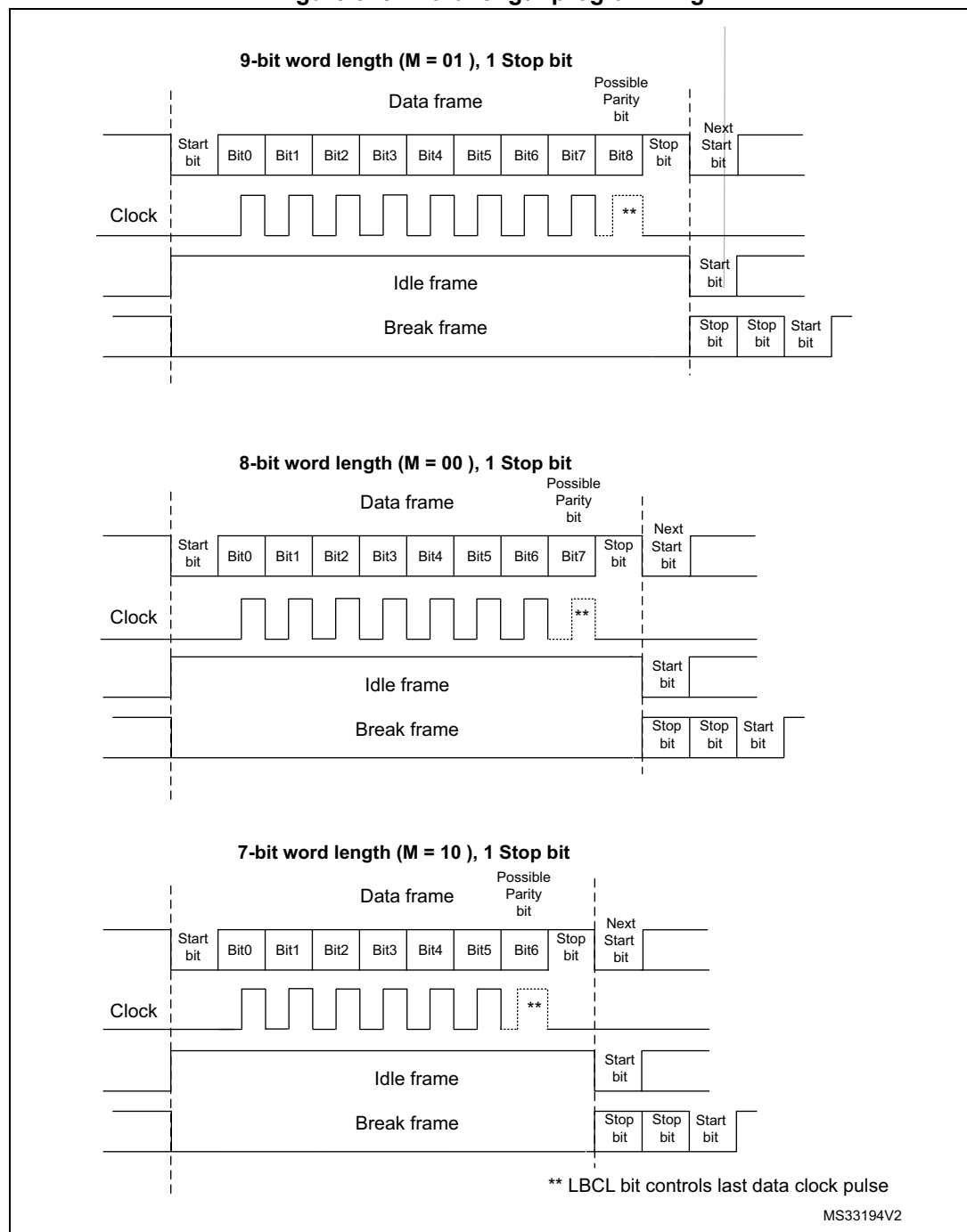
An **Idle character** is interpreted as an entire frame of "1"s (the number of "1"s includes the number of stop bits).

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 315. Word length programming**



### 33.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART\_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

*Note:* *The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.*

*The status flags are available in the USART\_ISR register.*

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART\_CR3 control register.

In this case:

- The RXFT flag is set in the USART\_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

RXFTCFG data have been received: one data in USART\_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART\_RDR). As a result, the next received data is not set the overrun flag.

- The TXFT flag is set in the USART\_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

### 33.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

#### Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART\_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART\_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

*Note:* The TE bit must be set before writing the data to be transmitted to the USART\_TDR. The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost. An idle frame is sent when the TE bit is enabled.

**Configurable stop bits**

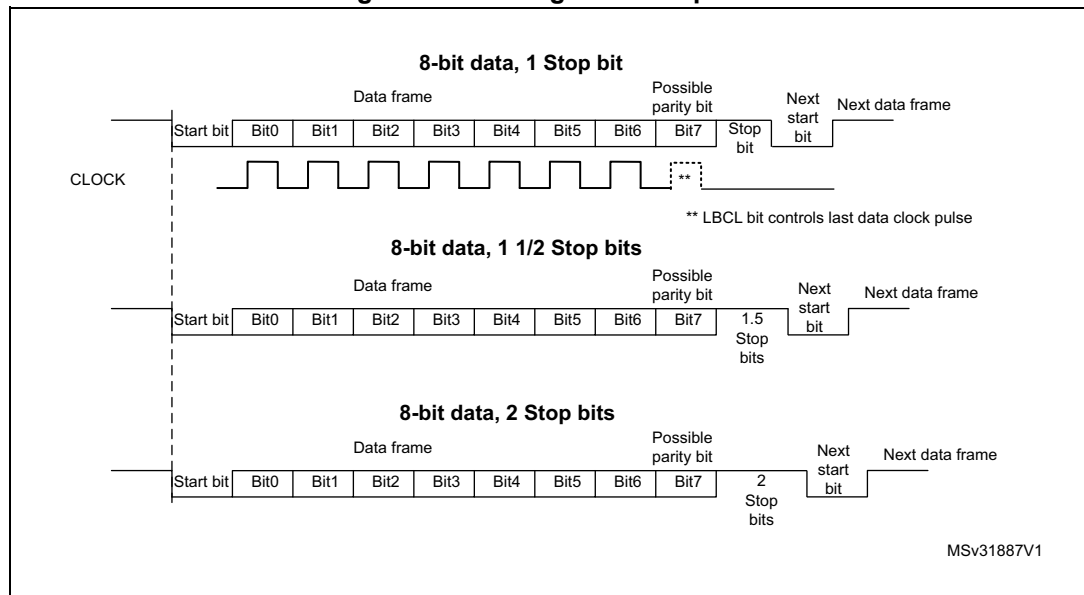
The number of stop bits to be transmitted with every character can be programmed in USART\_CR2, bits 13,12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This is supported by normal USART, single-wire and modem modes.
- **1.5 stop bits:** To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits (see [Figure 316](#)). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 316. Configurable stop bits**



**Character transmission procedure**

To transmit a character, follow the sequence below:

1. Program the M bits in USART\_CR1 to define the word length.
2. Select the desired baud rate using the USART\_BRR register.
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to 1.
5. Select DMA enable (DMAT) in USART\_CR3 if multibuffer communication must take place. Configure the DMA register as explained in [Section 33.5.19: Continuous communication using USART and DMA](#).
6. Set the TE bit in USART\_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART\_TDR register. Repeat this for each data to be transmitted in case of single buffer.
  - When FIFO mode is disabled, writing a data to the USART\_TDR clears the TXE flag.
  - When FIFO mode is enabled, writing a data to the USART\_TDR adds one data to the TXFIFO. Write operations to the USART\_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART\_TDR register, wait until TC = 1.
  - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
  - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

### Single byte communication

- When FIFO mode is disabled

Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:

- the data have been moved from the USART\_TDR register to the shift register and the data transmission has started;
- the USART\_TDR register is empty;
- the next data can be written to the USART\_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is ongoing, a write instruction to the USART\_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

When no transmission is ongoing, a write instruction to the USART\_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:

- the TXFIFO is not full;
- the USART\_TDR register is empty;
- the next data can be written to the USART\_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART\_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

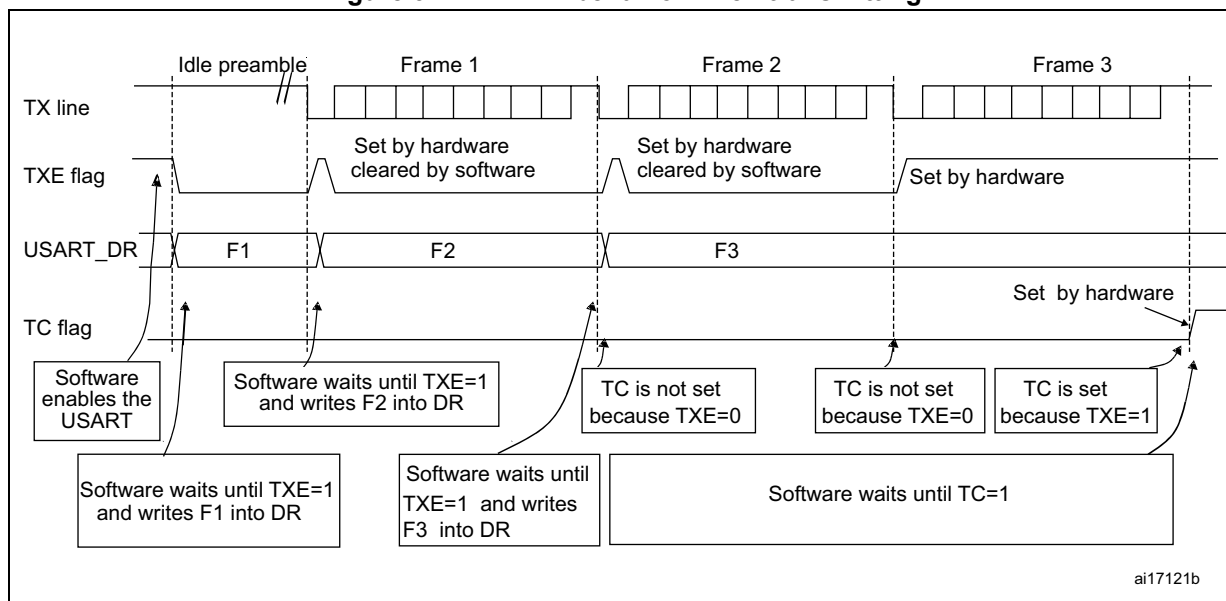
When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write operation to USART\_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART\_CR1 register.

After writing the last data to the USART\_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the device to enter the low-power mode (see [Figure 317: TC/TXE behavior when transmitting](#)).

Figure 317. 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 315).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

**Idle characters**

Setting the TE bit drives the USART to send an idle frame before the first data frame.

**33.5.6 USART receiver**

The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART\_CR1 register.

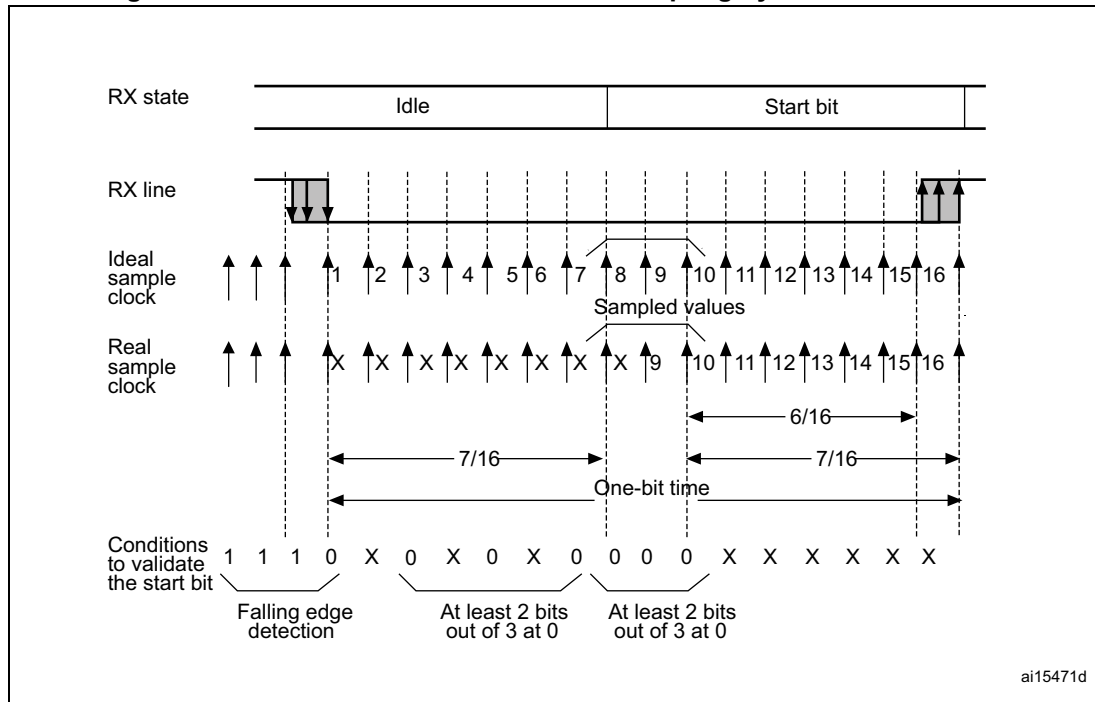
**Start bit detection**

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.



Figure 318. 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,

- a) for both samplings, 2 out of the 3 sampled bits are at '0' (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)
- or
- b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at '0'.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

## Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

### Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART\_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART\_BRR
3. Program the number of stop bits in USART\_CR2.
4. Enable the USART by writing the UE bit in USART\_CR1 register to '1'.
5. Select DMA enable (DMAR) in USART\_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 33.5.19: Continuous communication using USART and DMA](#).
6. Set the RE bit USART\_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART\_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART\_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to '1' in the USART\_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART\_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to '1' in USART\_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

### Break character

When a break character is received, the USART handles it as a framing error.

### Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

### Overrun error

- FIFO mode disabled
 

An overrun error occurs if a character is received and RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte reception.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available by reading the USART\_RDR register.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE or the EIE bit is set.
- FIFO mode enabled
 

An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

Data can not be transferred from the shift register to the USART\_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available by reading the USART\_RDR register.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART\_ICR register.

*Note: The ORE bit, when set, indicates that at least 1 data has been lost.*

*When the FIFO mode is disabled, there are two possibilities*

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

### Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section *Reset and clock control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

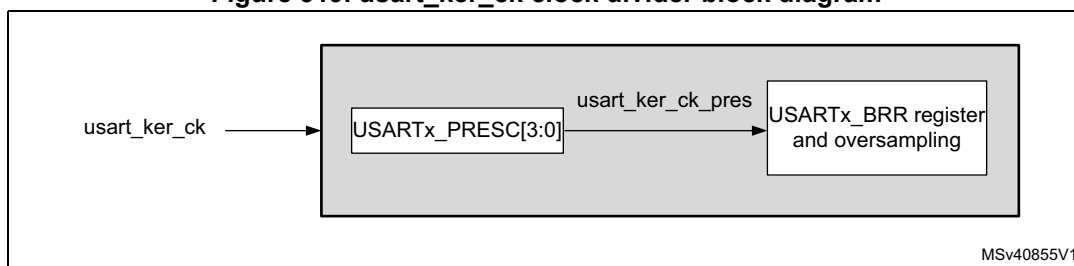
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is `usart_ker_ck`.

When the dual clock domain and the wake-up from low-power mode features are supported, the `usart_ker_ck` clock source can be configurable in the RCC (see Section *Reset and clock control (RCC)*). Otherwise the `usart_ker_ck` clock is the same as `usart_pclk`.

The `usart_ker_ck` clock can be divided by a programmable factor, defined in the `USARTx_PRESC` register.

Figure 319. `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 320](#) and [Figure 321](#)).

Depending on your application:

- select oversampling by 8 (`OVER8 = 1`) to achieve higher speed (up to `usart_ker_ck_pres/8`). In this case the maximum receiver tolerance to clock deviation is reduced (refer to [Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1004](#))
- select oversampling by 16 (`OVER8 = 0`) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum

$\text{usart\_ker\_ck\_pres}/16$  (where  $\text{usart\_ker\_ck\_pres}$  is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART\_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

Depending on your application:

- select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to [Figure 187](#)) because this indicates that a glitch occurred during the sampling.
- select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see [Section 33.5.8: Tolerance of the USART receiver to clock deviation on page 1004](#)). 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 and IrDA modes.

*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 320. Data sampling when oversampling by 16**

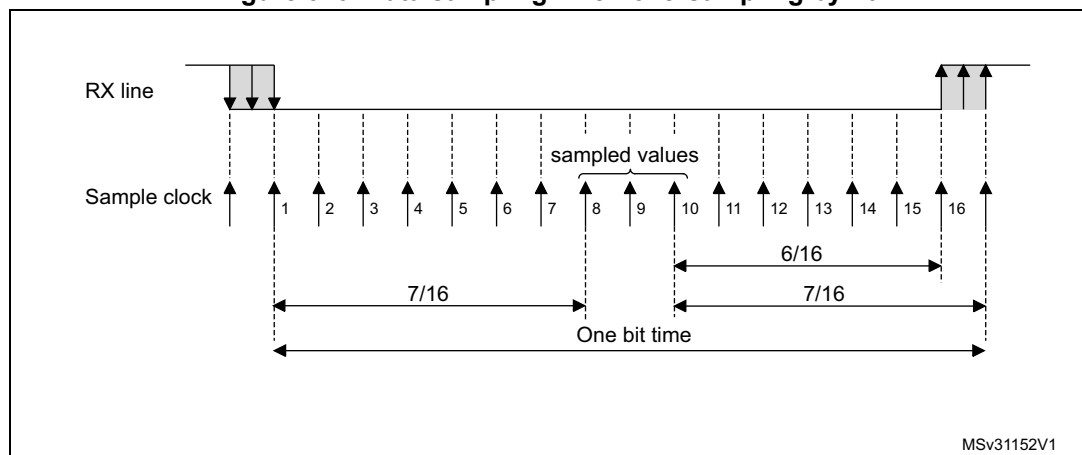


Figure 321. Data sampling when oversampling by 8

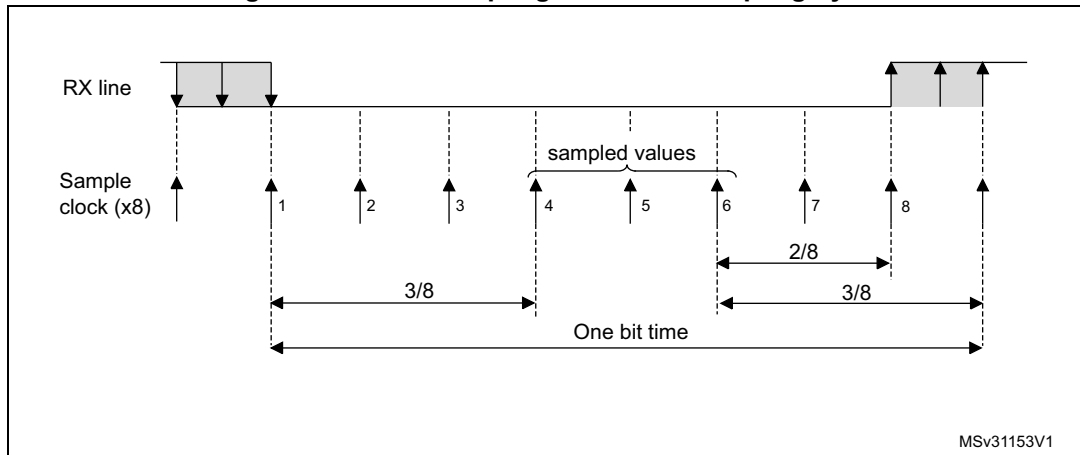


Table 187. Noise detection from sampled data

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 33.5.16: USART receiver timeout on page 1018](#) for more details).

- **2 stop bits**  
Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. The framing error flag is set if a framing error is detected during the first stop bit. The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

### 33.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART\_BRR register.

#### Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = '0' or '1')

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

#### Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

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

### 33.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.



The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$$

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 188](#), [Table 189](#), 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 188. 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 189. 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 188](#) and [Table 189](#) may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9-bit times when M = 10).

### 33.5.9 USART auto baud rate detection

The USART can detect and automatically set the USART\_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from  $\text{usart\_ker\_ck\_pres}/65535$  and  $\text{usart\_ker\_ck\_pres}/16$ .
- When oversampling by 8, the baud rate ranges from  $\text{usart\_ker\_ck\_pres}/65535$  and  $\text{usart\_ker\_ck\_pres}/8$ .

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART\_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0:** Any character starting with a bit at '1'.  
In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.  
In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2:** A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3:** A 0x55 character frame.  
In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART\_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART\_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART\_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a '0').

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART\_ISR register.

*Note:* The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

### 33.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART\_CR1 register.

*Note:* When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two `usart_ker_ck` cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART\_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART\_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART\_CR1 register:

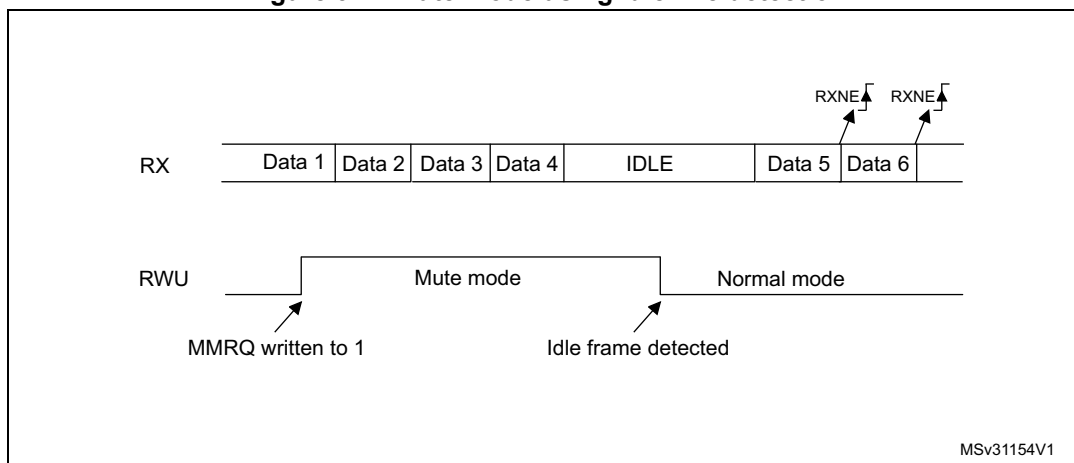
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

#### Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to '1' and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART\_ISR register. An example of mute mode behavior using Idle line detection is given in [Figure 322](#).

Figure 322. 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.

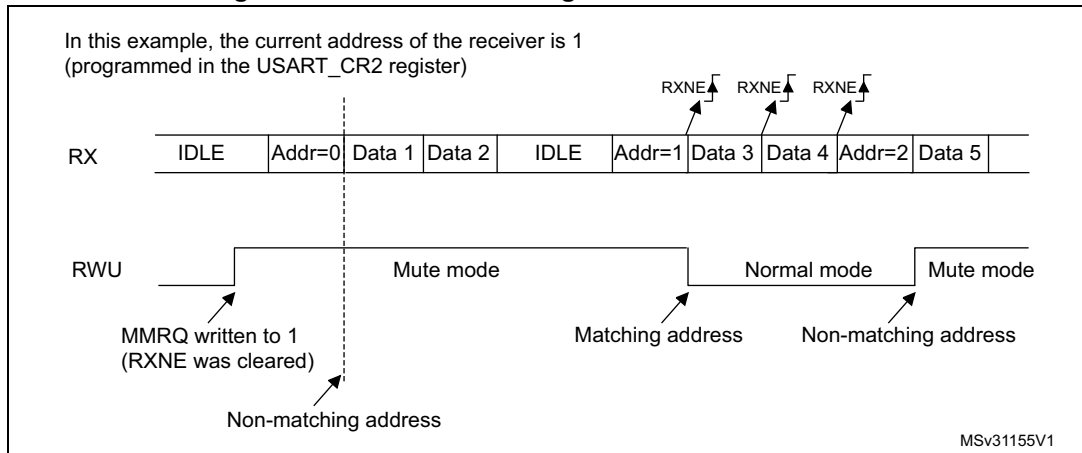
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 323](#).

**Figure 323. Mute mode using address mark detection**



### 33.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

#### Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART\_CR2 register and the RTOIE in the USART\_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

#### Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

### 33.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART\_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in [Table 190](#).

**Table 190. 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

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

#### Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits are set, the parity bit is equal to 0 if even parity is selected (PS bit in USART\_CR1 = 0).

#### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART\_CR1 = 1).

#### Parity checking in reception

If the parity check fails, the PE flag is set in the USART\_ISR register and an interrupt is generated if PEIE is set in the USART\_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART\_ICR register.

#### Parity generation in transmission

If the PCE bit is set in USART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS=1).

### 33.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to [Section 33.4: USART implementation on page 986](#).

The LIN mode is selected by setting the LINEN bit in the USART\_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART\_CR2 register,
- SCEN, HDSEL and IREN in the USART\_CR3 register.

#### LIN transmission

The procedure described in [Section 33.5.4](#) has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 '0 bits as a break character. Then two bits of value '1 are sent to enable the next start detection.

#### LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART\_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART\_CR2) or 11 (when LBDL = 1 in USART\_CR2) consecutive bits are detected as '0, and are followed by a delimiter character, the LBDF flag is set in USART\_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a '1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (i.e. stop bit detected at '0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a '1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the [Figure 324: Break detection in LIN mode \(11-bit break length - LBDL bit is set\) on page 1013](#).

Examples of break frames are given on [Figure 325: Break detection in LIN mode vs. Framing error detection on page 1014](#).



**Figure 324. Break detection in LIN mode (11-bit break length - LBDL bit is set)**

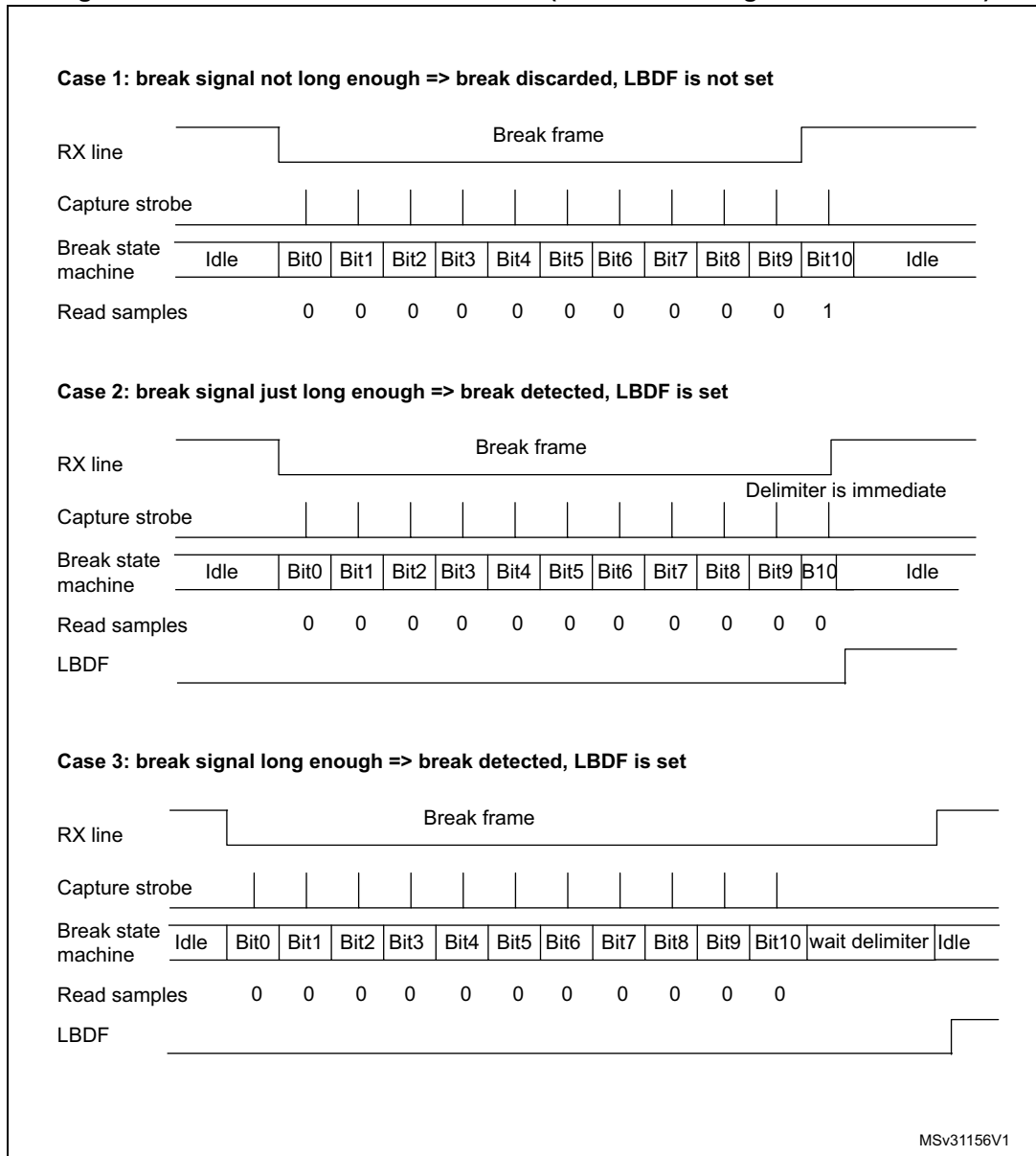
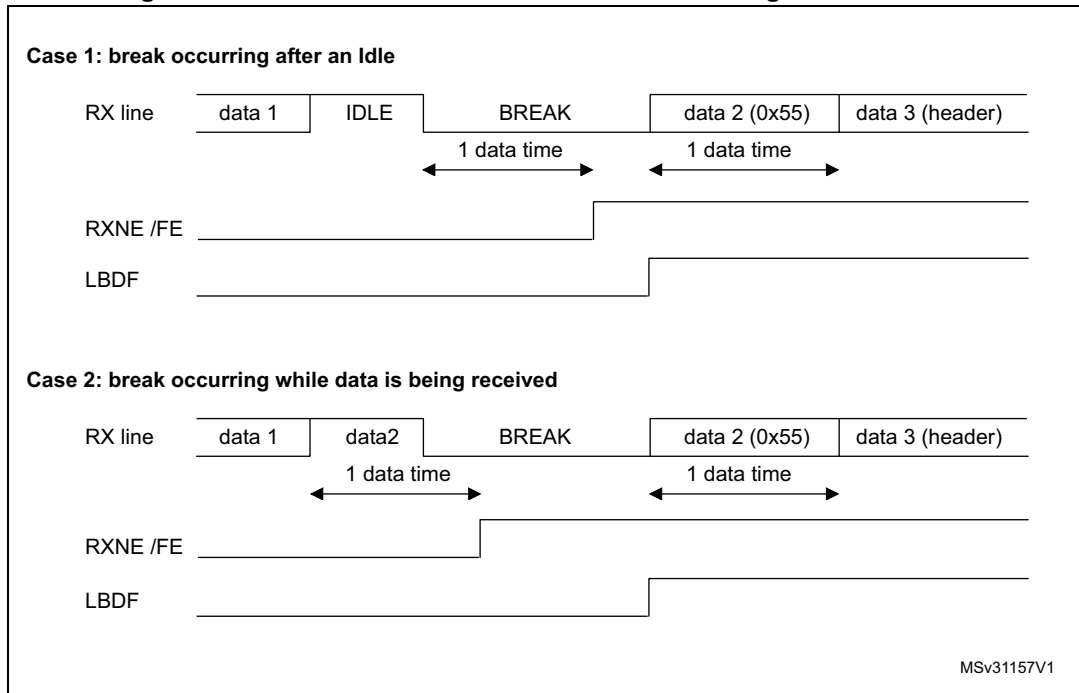


Figure 325. Break detection in LIN mode vs. Framing error detection



### 33.5.14 USART synchronous mode

#### Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART\_CR2 register to '1'. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART\_CR2 register,
- SCEN, HDSEL and IREN bits in the USART\_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART\_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART\_CR2 register is used to select the clock polarity, and the CPHA bit in the USART\_CR2 register is used to select the phase of the external clock (see [Figure 326](#), [Figure 327](#) and [Figure 328](#)).

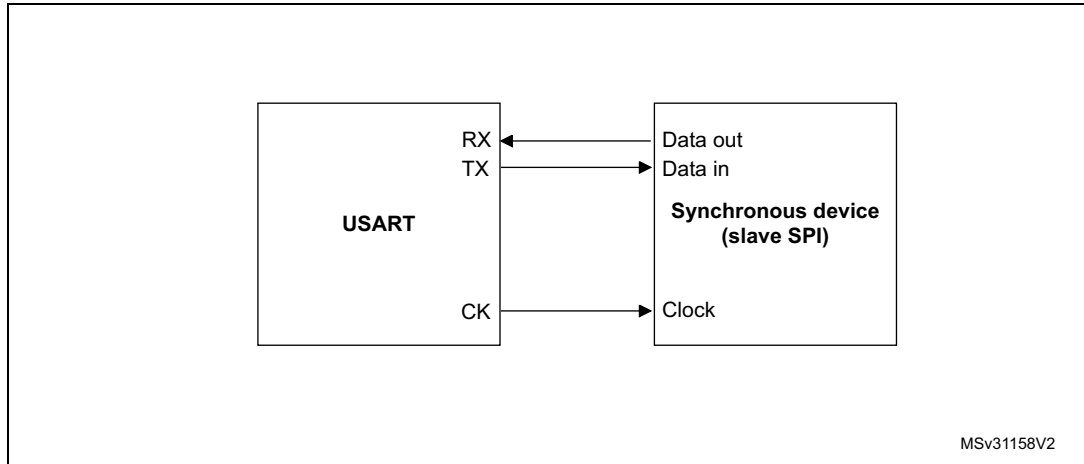
During the Idle state, preamble and send break, the external CK clock is not activated.

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.

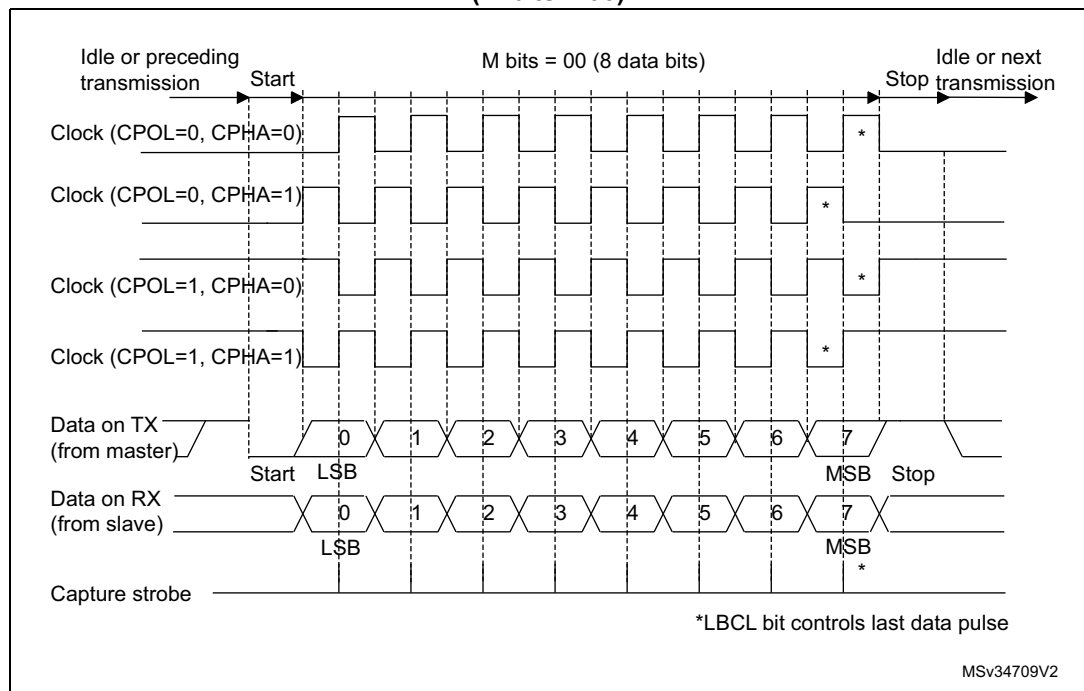
In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the 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.

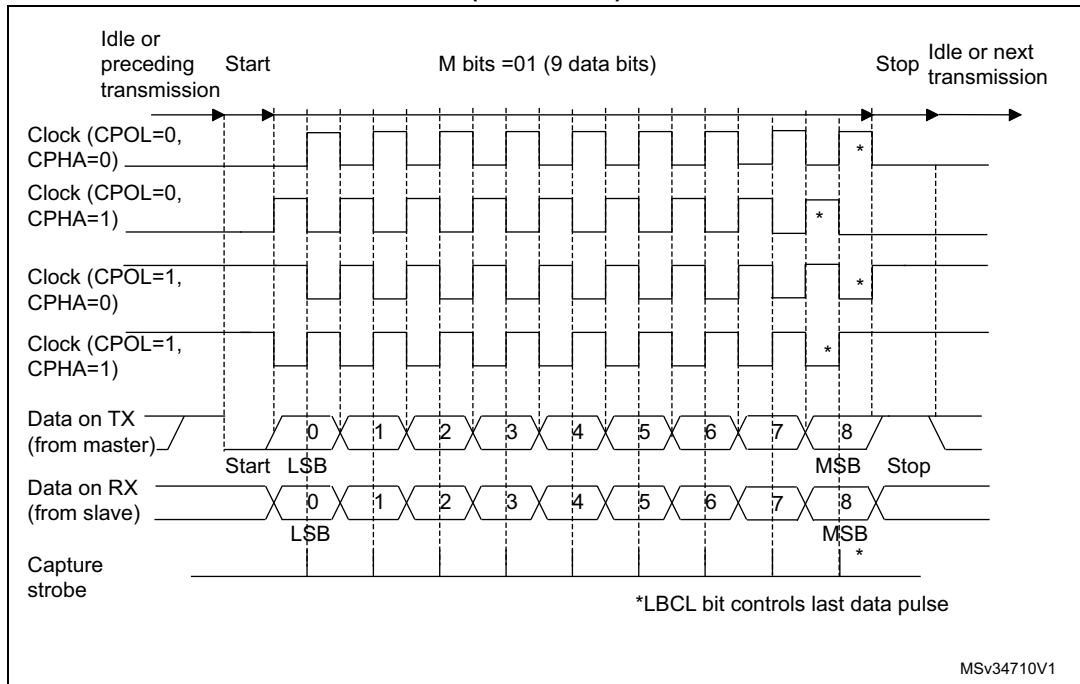
**Figure 326. USART example of synchronous master transmission**



**Figure 327. USART data clock timing diagram in synchronous master mode (M bits = 00)**



**Figure 328. USART data clock timing diagram in synchronous master mode (M bits = 01)**



**Slave mode**

The synchronous slave mode is selected by programming the SLVEN bit in the USART\_CR2 register to '1'. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART\_CR2 register,
- SCEN, HDSEL and IREN bits in the USART\_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

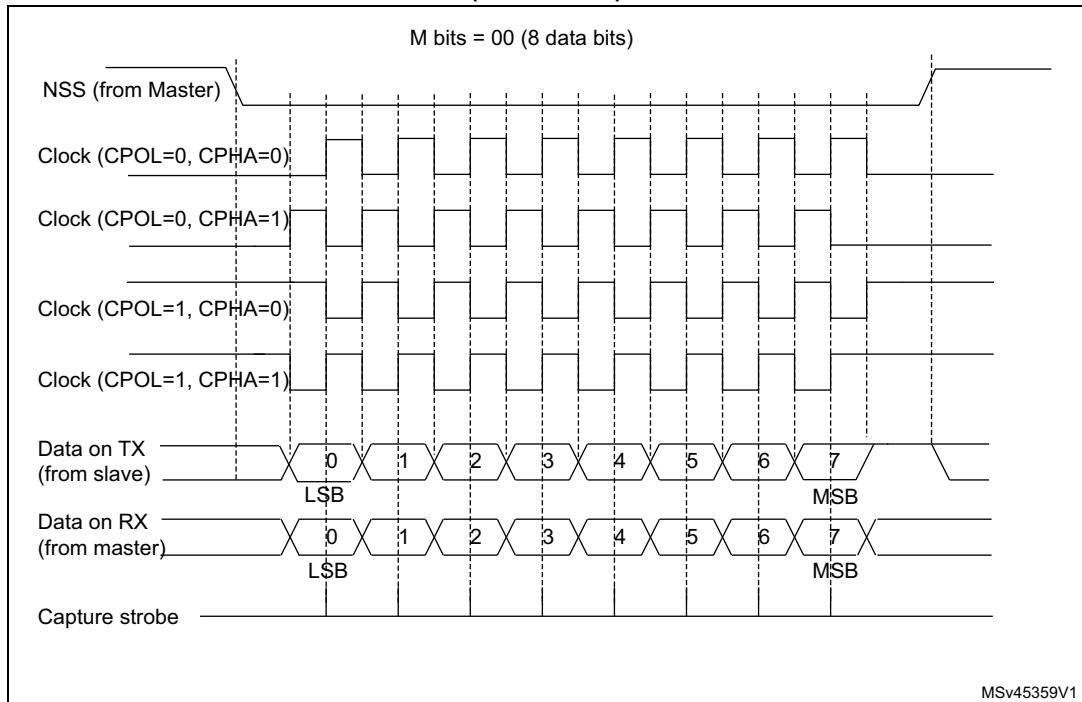
*Note:* When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (usart\_ker\_ck\_pres) must be greater than 3 times the CK input frequency.

The CPOL bit and the CPHA bit in the USART\_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see [Figure 329](#)).

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 329. 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.

### 33.5.15 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART\_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART\_CR2 register,
- SCEN and IREN bits in the USART\_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART\_CR3.

As soon as HDSEL is written to '1':

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

### 33.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART\_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART\_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = '00' or STOP = '11'
- from the end of the second stop bit if STOP = '10'.
- from the beginning of the stop bit if STOP = '01'.

When the timeout duration has elapsed, the RTOF flag in the USART\_ISR register is set. A timeout is generated if RTOIE bit in USART\_CR1 register is set.

### 33.5.17 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to [Section 33.4: USART implementation on page 986](#).

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.

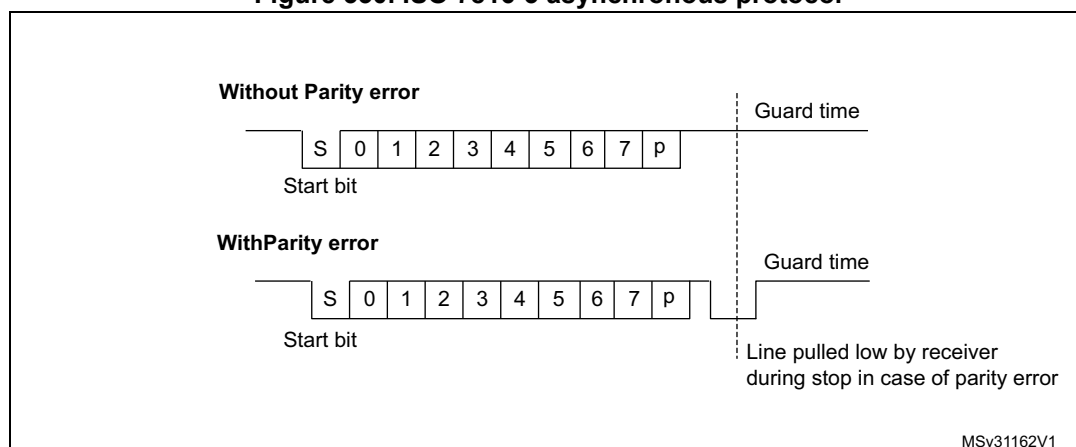
The USART should be configured as:

- 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 330](#) shows examples of what can be seen on the data line with and without parity error.

**Figure 330. 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.

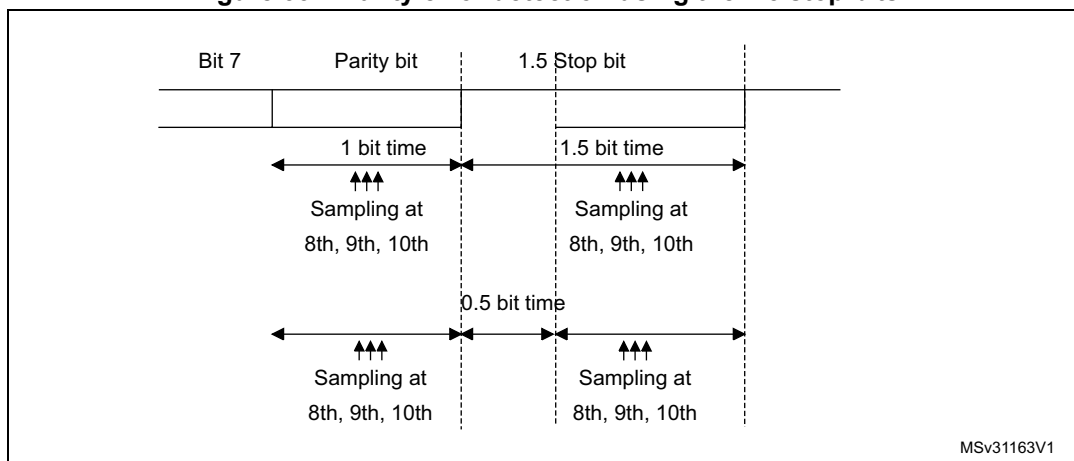
*Note:* Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

*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 331* 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 331. 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 EOBFF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

*Note:* The error checking code (LRC/CRC) must be computed/verified by software.

### Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV = 1.

*Note:* When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, assuming that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH results in a USART received character of 03 and an odd parity.

Therefore, two methods are available for TS pattern recognition:

#### Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART.

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

#### Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

### 33.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to [Section 33.4: USART implementation on page 986](#).

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 332](#)).

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 333](#)).
- 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 ( $1.42 \text{ MHz} < \text{PSC} < 2.12 \text{ MHz}$ ). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver  
Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/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 332. IrDA SIR ENDEC block diagram

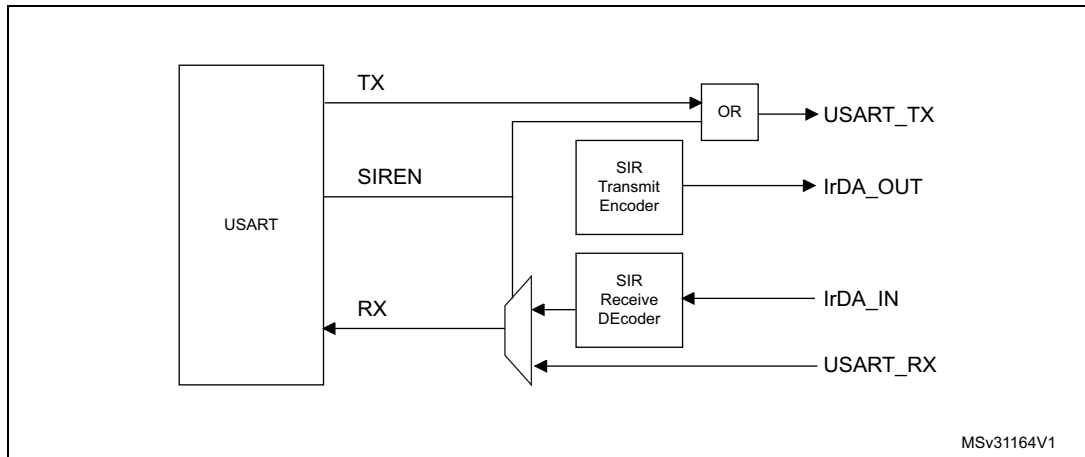
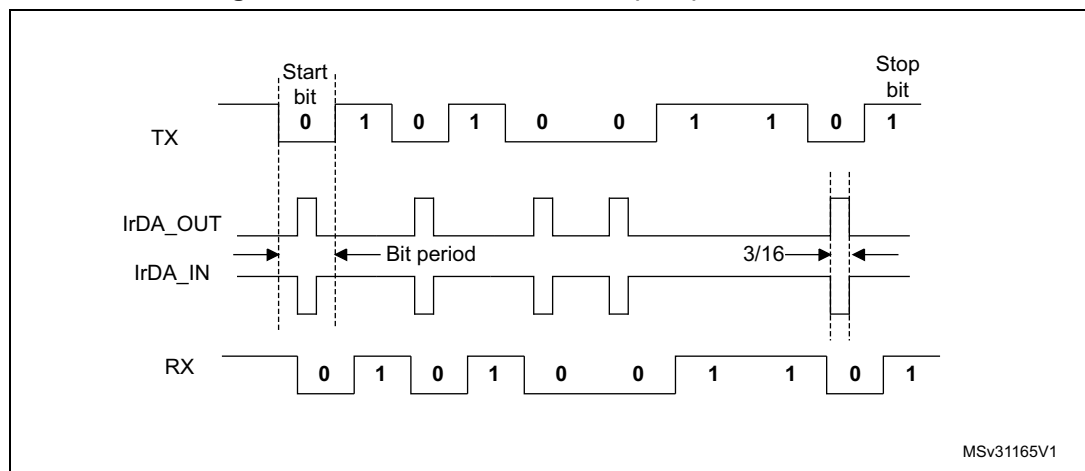


Figure 333. IrDA data modulation (3/16) - normal mode



### 33.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

*Note:* Refer to [Section 33.4: USART implementation on page 986](#) to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in [Section 33.5.6](#). To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART\_ISR register.

#### Transmission using DMA

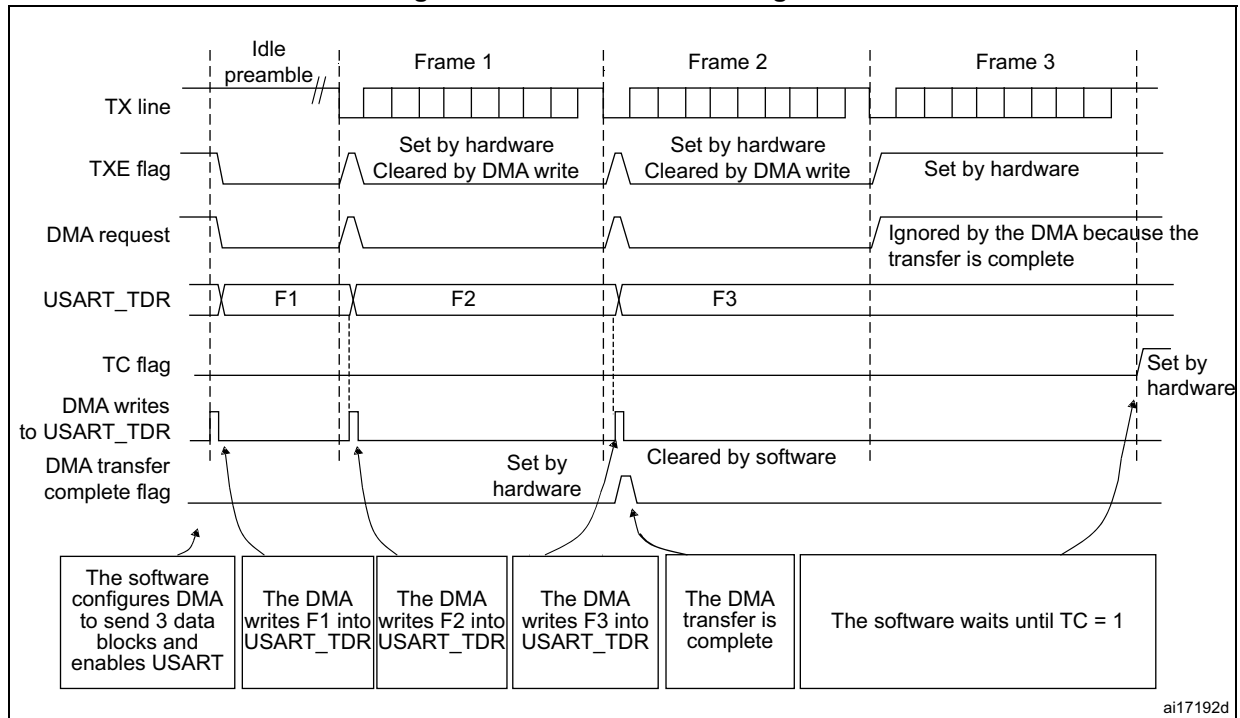
DMA mode can be enabled for transmission by setting DMAT bit in the USART\_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART\_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART\_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART\_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART\_ISR register by setting the TCCF bit in the USART\_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 334. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

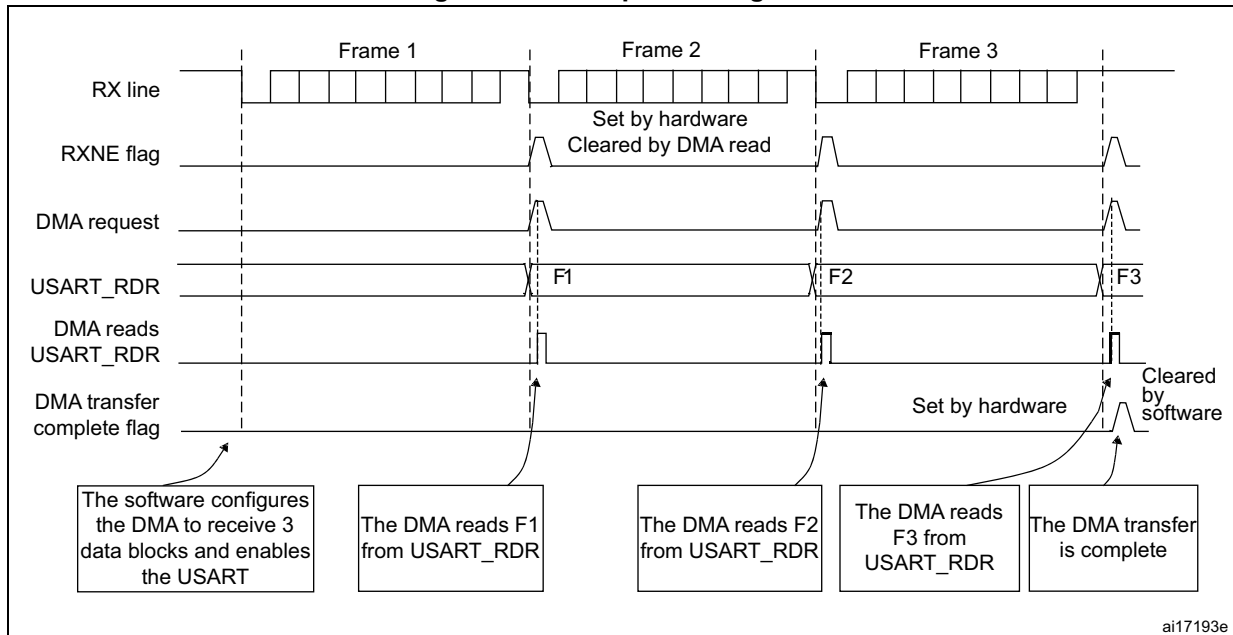
### Reception using DMA

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 335. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

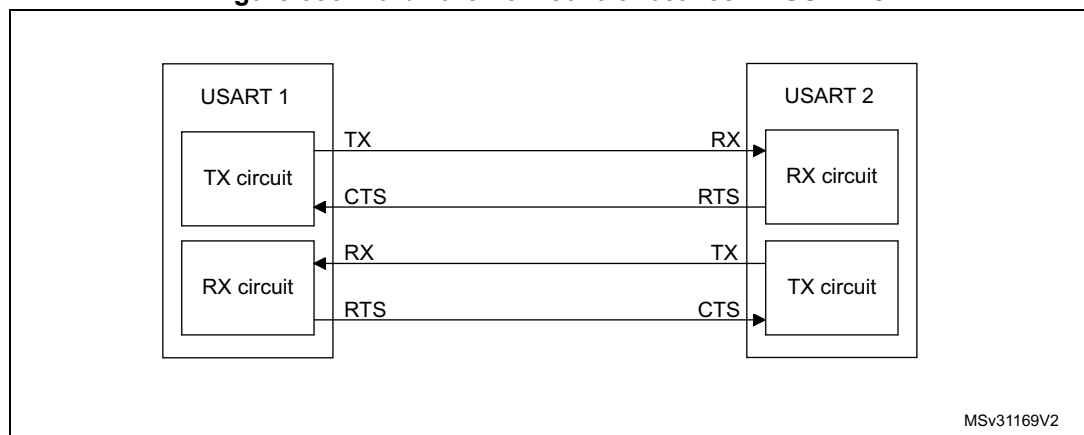
**Error flagging and interrupt generation in multibuffer communication**

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.

**33.5.20 RS232 hardware flow control and RS485 Driver Enable**

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The Figure 336 shows how to connect 2 devices in this mode:

Figure 336. Hardware flow control between 2 USARTs



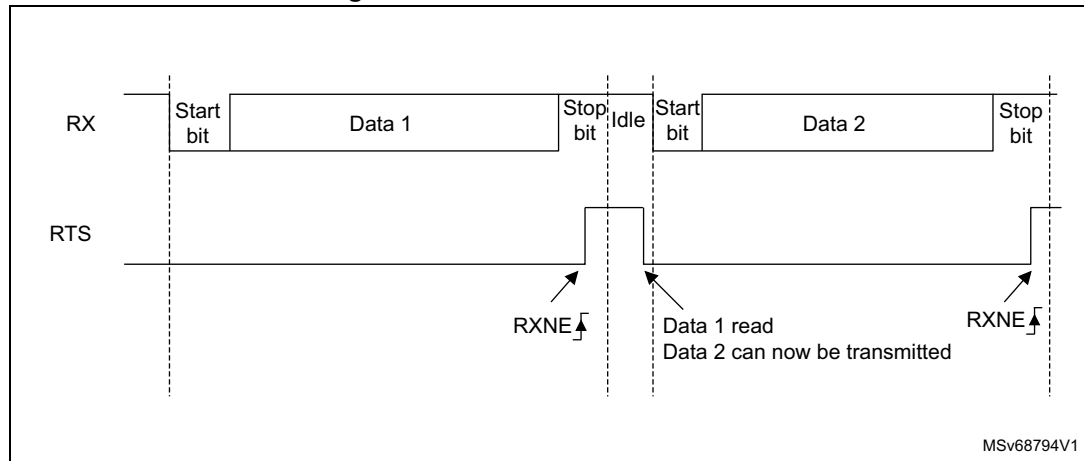


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to '1' in the USART\_CR3 register.

**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 337](#) shows an example of communication with RTS flow control enabled.

**Figure 337. RS232 RTS flow control**



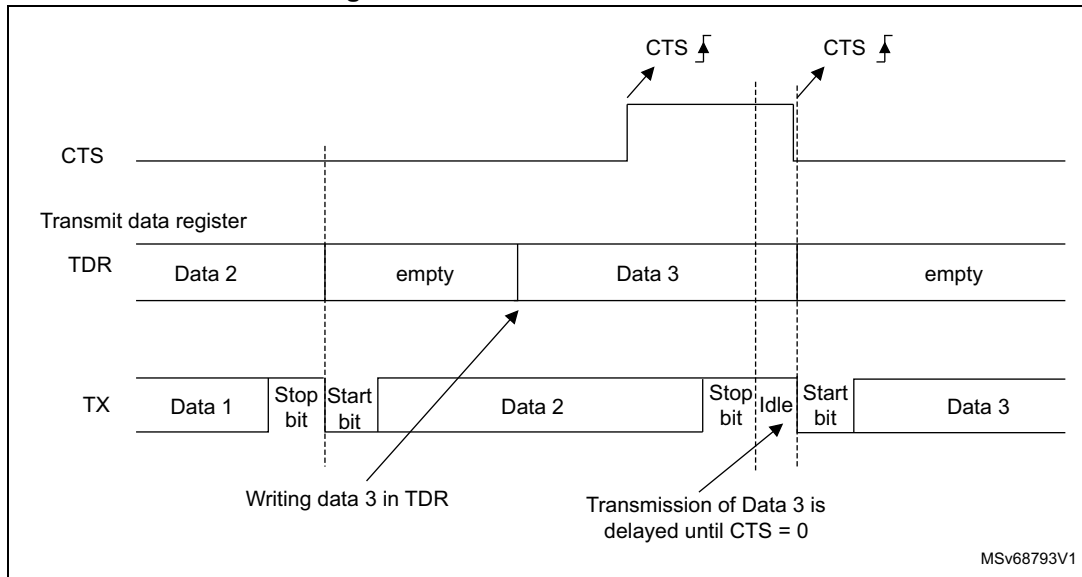
*Note:* When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

**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 338](#) shows an example of communication with CTS flow control enabled.

Figure 338. 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).

### 33.5.21 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the `usart_pclk` clock is disabled.

The USART is able to wake up the MCU from low-power mode when the `UESM` bit is set.

When the `usart_pclk` is gated, the USART provides a wake-up interrupt (`usart_wkup`) if a specific action requiring the activation of the `usart_pclk` clock is needed:

- If FIFO mode is disabled
  - `usart_pclk` clock has to be activated to empty the USART data register.
  - In this case, the `usart_wkup` interrupt source is `RXNE` set to '1'. The `RXNEIE` bit must be set before entering low-power mode.
- If FIFO mode is enabled
  - `usart_pclk` clock has to be activated to:
    - to fill the `TXFIFO`
    - or to empty the `RXFIFO`
  - In this case, the `usart_wkup` interrupt source can be:
    - `RXFIFO` not empty. In this case, the `RXFNEIE` bit must be set before entering low-power mode.
    - `RXFIFO` full. In this case, the `RXFFIE` bit must be set before entering low-power mode, the number of received data corresponds to the `RXFIFO` size, and the `RXFF` flag is not set.
    - `TXFIFO` empty. In this case, the `TXFEIE` bit must be set before entering low-power mode.

This enables sending/receiving the data in the `TXFIFO/RXFIFO` during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the `usart_wkup` interrupt source can be one of the following events:

- `TXFIFO` threshold reached. In this case, the `TXFTIE` bit must be set before entering low-power mode.
- `RXFIFO` threshold reached. In this case, the `RXFTIE` bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum `RXFIFO` size if the wake-up time is less than the time required to receive a single byte across the line.

Using the `RXFIFO` full, `TXFIFO` empty, `RXFIFO` not empty and `RXFIFO/TXFIFO` threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific `usart_wkup` interrupt can be selected through the `WUS` bitfields.

When the wake-up event is detected, the `WUF` flag is set by hardware and a `usart_wkup` interrupt is generated if the `WUFIE` bit is set.

*Note:* Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the *BUSY* flag cannot ensure that low-power mode is never entered when data reception is ongoing.

*The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.*

*When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.*

*When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.*

*When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.*

### Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the *RXNE* flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

*Note:* When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (i.e. the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

### Wake-up from low-power mode when USART kernel clock (*usart\_ker\_ck*) is OFF in low-power mode

If during low-power mode, the *usart\_ker\_ck* clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the *usart\_ker\_ck* clock to be switched ON thanks to the *usart\_ker\_ck\_req* signal. *usart\_ker\_ck* is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, *usart\_ker\_ck* is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 339 shows the USART behavior when the wake-up event is verified.

**Figure 339. Wake-up event verified (wake-up event = address match, FIFO disabled)**

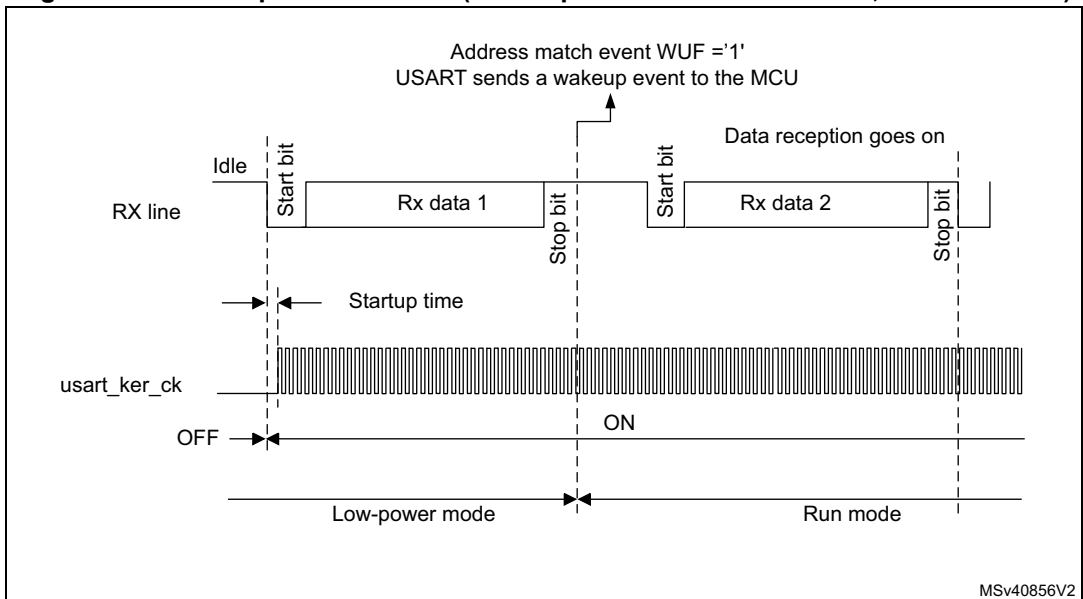
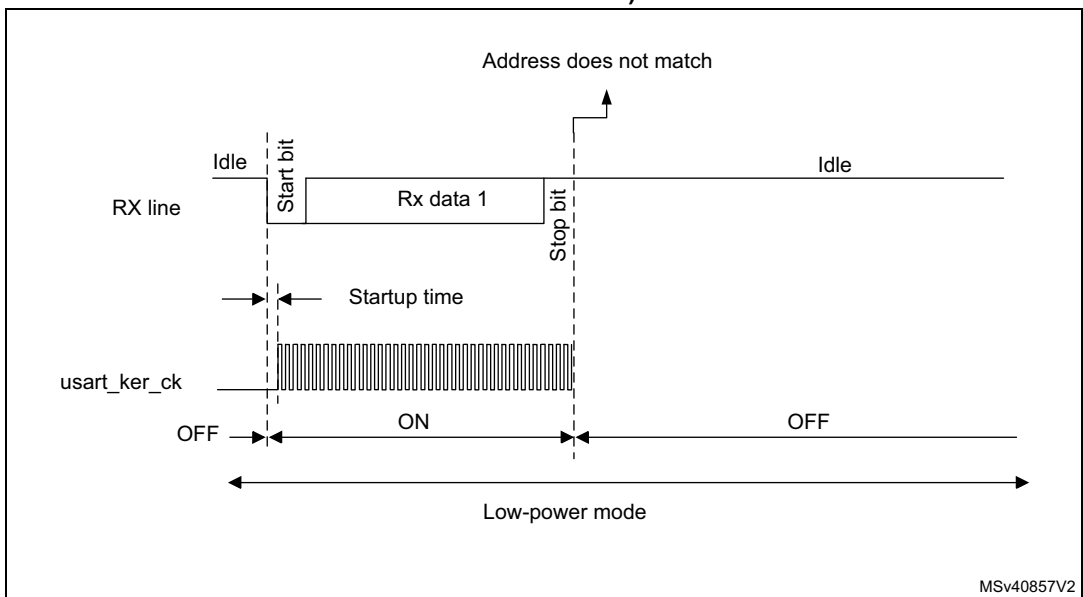


Figure 340 shows the USART behavior when the wake-up event is not verified.

**Figure 340. 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 33.5.8: Tolerance of the USART receiver to clock deviation](#)).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to [Table 188: Tolerance of the USART receiver when BRR \[3:0\] = 0000](#), the USART receiver tolerance equals 3.41%.

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{USART receiver tolerance}$$

$$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit\ Min})$$

$$T_{bit\ Min} = t_{WUUSART} / (11 \times D_{WUmax})$$

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 usart\_ker\_ck inaccuracy.

For example, if HSI is used as usart\_ker\_ck, and the HSI 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).

$$D_{WUmax} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit\ min} = 3 \mu s / (11 \times 2.41\%) = 11.32 \mu s.$$

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is:  $1/11.32 \mu s = 88.36 \text{ Kbaud}$ .

## 33.6 USART in low-power modes

Table 191. 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.

1. Refer to [Section 33.4: USART implementation](#) to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

### 33.7 USART interrupts

Refer to [Table 192](#) for a detailed description of all USART interrupt requests.

**Table 192. 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	TXFNIE	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	TCBGIE	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	
	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			

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to [Section 33.4: USART implementation](#) for the list of supported Stop modes.



2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART\_RDR. In Stop mode, USART\_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

### 33.8 USART registers

Refer to [Section 1.2 on page 55](#) for a list of abbreviations used in register descriptions.  
 The peripheral registers have to be accessed by words (32 bits).

#### 33.8.1 USART control register 1 (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

##### FIFO mode enabled

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	TXFNFI E	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 EOBIF flag is set in the USART\_ISR register

*Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. [Section 33.4: USART implementation on page 986](#).*

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

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 33.4: USART implementation on page 986](#).*

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART\_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

*In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.*

### 33.8.2 USART control register 1 [alternate] (USART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

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 Bblock interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBIF flag is set in the USART\_ISR register

*Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 986.*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART\_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 33.4: USART implementation on page 986.*

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 33.4: USART implementation on page 986.*

- Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time  
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).  
If the USART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.  
This bitfield can only be written when the USART is disabled (UE = 0).  
*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 15 **OVER8**: Oversampling mode  
0: Oversampling by 16  
1: Oversampling by 8  
This bit can only be written when the USART is disabled (UE = 0).  
*Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.*
- Bit 14 **CMIE**: Character match interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated when the CMF bit is set in the USART\_ISR register.
- Bit 13 **MME**: Mute mode enable  
This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.  
0: Receiver in active mode permanently  
1: Receiver can switch between mute mode and active mode.
- Bit 12 **M0**: Word length  
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).  
This bit can only be written when the USART is disabled (UE = 0).
- Bit 11 **WAKE**: Receiver wake-up method  
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.  
0: Idle line  
1: Address mark  
This bitfield can only be written when the USART is disabled (UE = 0).
- Bit 10 **PCE**: Parity control enable  
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).  
0: Parity control disabled  
1: Parity control enabled  
This bitfield can only be written when the USART is disabled (UE = 0).
- Bit 9 **PS**: Parity selection  
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.  
0: Even parity  
1: Odd parity  
This bitfield can only be written when the USART is disabled (UE = 0).

- Bit 8 **PEIE**: PE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever PE = 1 in the USART\_ISR register
- Bit 7 **TXEIE**: Transmit data register empty  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TXE =1 in the USART\_ISR register
- Bit 6 **TCIE**: Transmission complete interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TC = 1 in the USART\_ISR register
- Bit 5 **RXNEIE**: Receive data register not empty  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART\_ISR register
- Bit 4 **IDLEIE**: IDLE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever IDLE = 1 in the USART\_ISR register
- Bit 3 **TE**: Transmitter enable  
This bit enables the transmitter. It is set and cleared by software.  
0: Transmitter is disabled  
1: Transmitter is enabled
- Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART\_ISR register.*
- In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.*
- Bit 2 **RE**: Receiver enable  
This bit enables the receiver. It is set and cleared by software.  
0: Receiver is disabled  
1: Receiver is enabled and begins searching for a start bit
- Bit 1 **UESM**: USART enable in low-power mode  
When this bit is cleared, the USART cannot wake up the MCU from low-power mode.  
When this bit is set, the USART can wake up the MCU from low-power mode.  
This bit is set and cleared by software.  
0: USART not able to wake up the MCU from low-power mode.  
1: USART able to wake up the MCU from low-power mode.
- Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.*
- If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART\_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

*In smartcard mode, (SCEN = 1), the CK pin is always available when CLKEN = 1, regardless of the UE bit value.*

### 33.8.3 USART control register 2 (USART\_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31								30								29								28								27								26								25								24								23								22								21								20								19								18								17								16							
ADD[7:0]																RTOEN				ABRMOD[1:0]				ABREN				MSBFIRST				DATAINV				TXINV				RXINV																																																																																							
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				DISNSS				Res.				Res.				SLVEN																																																																							
rw				rw				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 33.4: USART implementation on page 986](#).*

**Bits 22:21 ABRMOD[1:0]:** Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 and Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

*Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)*

*If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 20 ABREN:** Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 19 MSBFIRST:** Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 18 DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 17 TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels ( $V_{DD}$  = 1/idle, Gnd = 0/mark)

1: TX pin signal values are inverted ( $V_{DD}$  = 0/mark, Gnd = 1/idle).

This 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/idle, Gnd = 0/mark)

1: RX pin signal values are inverted ( $V_{DD}$  = 0/mark, Gnd = 1/idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).



**Bit 15 SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 14 LINEN:** LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART\_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value.*

*Refer to [Section 33.4: USART implementation on page 986](#).*

**Bits 13:12 STOP[1:0]:** Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

**Bit 11 CLKEN:** Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit can only be written when the USART is disabled (UE = 0).

*Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

*In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:*

*UE = 0*

*SCEN = 1*

*GTPR configuration*

*CLKEN = 1*

*UE = 1*

**Bit 10 CPOL:** Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

*Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value.*

*Refer to [Section 33.4: USART implementation on page 986](#).*

- 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 320](#) and [Figure 321](#))  
0: The first clock transition is the first data capture edge  
1: The second clock transition is the first data capture edge  
This bit can only be written when the USART is disabled (UE = 0).  
*Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 8 **LBCL**: Last bit clock pulse  
This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.  
0: The clock pulse of the last data bit is not output to the CK pin  
1: The clock pulse of the last data bit is output to the CK pin  
**Caution:** The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART\_CR1 register.  
This bit can only be written when the USART is disabled (UE = 0).  
*Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 7 Reserved, must be kept at reset value.
- Bit 6 **LBDIE**: LIN break detection interrupt enable  
Break interrupt mask (break detection using break delimiter).  
0: Interrupt is inhibited  
1: An interrupt is generated whenever LBDF = 1 in the USART\_ISR register  
*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 5 **LBDL**: LIN break detection length  
This bit is for selection between 11 bit or 10 bit break detection.  
0: 10-bit break detection  
1: 11-bit break detection  
This bit can only be written when the USART is disabled (UE = 0).  
*Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection  
This bit is for selection between 4-bit address detection or 7-bit address detection.  
0: 4-bit address detection  
1: 7-bit address detection (in 8-bit data mode)  
This bit can only be written when the USART is disabled (UE = 0)  
*Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.*
- Bit 3 **DIS\_NSS**: NSS pin enable  
When the DIS\_NSS bit is set, the NSS pin input is ignored.  
0: SPI slave selection depends on NSS input pin.  
1: SPI slave is always selected and NSS input pin is ignored.  
*Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.

1: Slave mode enabled.

*Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value.*

*Refer to [Section 33.4: USART implementation on page 986](#).*

*Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.*

### 33.8.4 USART control register 3 (USART\_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31			30			29			28			27			26			25			24			23			22			21			20			19			18			17			16		
TXFTCFG[2:0]			RXFTIE			RXFTCFG[2:0]			TCBGTIE			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			OVRDIS			ONEBIT			CTSIE			CTSE			RTSE			DMAT			DMAR			SCEN			NACK			HDSSEL			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 33.4: USART implementation on page 986](#).*

- 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 33.4: USART implementation on page 986](#).*
- Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection  
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).  
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)  
01: Reserved.  
10: WUF active on start bit detection  
11: WUF active on RXNE/RXFNE.  
This bitfield can only be written when the USART is disabled (UE = 0).  
*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count  
This bitfield specifies the number of retries for transmission and reception in smartcard mode.  
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).  
In reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).  
This bitfield must be programmed only when the USART is disabled (UE = 0).  
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.  
0x0: retransmission disabled - No automatic retransmission in transmit mode.  
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)  
*Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 16 Reserved, must be kept at reset value.
- Bit 15 **DEP**: Driver enable polarity selection  
0: DE signal is active high.  
1: DE signal is active low.  
This bit can only be written when the USART is disabled (UE = 0).  
*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

- Bit 14 **DEM**: Driver enable mode  
This bit enables the user to activate the external transceiver control, through the DE signal.  
0: DE function is disabled.  
1: DE function is enabled. The DE signal is output on the RTS pin.  
This bit can only be written when the USART is disabled (UE = 0).  
*Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. [Section 33.4: USART implementation on page 986](#).*
- 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 33.4: USART implementation on page 986](#).*
- 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 33.4: USART implementation on page 986](#).*

**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 33.4: USART implementation on page 986](#).*

**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 33.4: USART implementation on page 986](#).*

**Bit 4 NACK:** Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 3 HDSEL:** Half-duplex selection

Selection of single-wire half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

**Bit 2 IRLP:** IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 1 IREN:** IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

*Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART\_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART\_ISR register.

### 33.8.5 USART baud rate register (USART\_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

### 33.8.6 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.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The Transmission Complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

Bits 7:0 **PSC[7:0]**: Prescaler value

**In IrDA low-power and normal IrDA mode:**

PSC[7:0] = IrDA normal and low-power baud rate

PSC[7:0] is used to program the prescaler for dividing the USART source clock to achieve the low-power frequency: the source clock is divided by the value given in the register (8 significant bits):

**In smartcard mode:**

PSC[4:0] = Prescaler value

PSC[4:0] is used to program the prescaler for dividing the USART source clock to provide the smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: Divides the source clock by 1 (IrDA mode) / by 2 (smartcard mode)

00010: Divides the source clock by 2 (IrDA mode) / by 4 (smartcard mode)

00011: Divides the source clock by 3 (IrDA mode) / by 6 (smartcard mode)

...

11111: Divides the source clock by 31 (IrDA mode) / by 62 (smartcard mode)

0010 0000: Divides the source clock by 32 (IrDA mode)

...

1111 1111: Divides the source clock by 255 (IrDA mode)

This bitfield can only be written when the USART is disabled (UE = 0).

*Note: Bits [7:5] must be kept cleared if smartcard mode is used.*

*This bitfield is reserved and forced by hardware to '0' when the smartcard and IrDA modes are not supported. Refer to [Section 33.4: USART implementation on page 986](#).*

### 33.8.7 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]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w



Bits 31:24 **BLLEN[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:

BLLEN = 0: 0 information characters + LEC

BLLEN = 1: 0 information characters + CRC

BLLEN = 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: RTOF can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.*

*This register is reserved and forced by hardware to “0x00000000” when the Receiver timeout feature is not supported. Refer to [Section 33.4: USART implementation on page 986](#).*

### 33.8.8 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.	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.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRR Q
											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 33.4: USART implementation on page 986.*

### 33.8.9 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 33.4: USART implementation on page 986](#).*

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART\_RDR register).

An interrupt is generated if the RXFFIE bit = 1 in the USART\_CR1 register.

0: RXFIFO not full.

1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART\_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART\_CR1 register.

0: TXFIFO not empty.

1: TXFIFO empty.

- Bit 22 **REACK**: Receive enable acknowledge flag  
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.  
It can be used to verify that the USART is ready for reception before entering low-power mode.  
*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 21 **TEACK**: Transmit enable acknowledge flag  
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.  
It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART\_CR1 register, in order to respect the TE = 0 minimum period.
- Bit 20 **WUF**: Wake-up from low-power mode flag  
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART\_ICR register.  
An interrupt is generated if WUFIE = 1 in the USART\_CR3 register.  
*Note: When UESM is cleared, WUF flag is also cleared.*  
*If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 19 **RWU**: Receiver wake-up from mute mode  
This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART\_CR1 register.  
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART\_RQR register.  
0: Receiver in active mode  
1: Receiver in mute mode  
*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*
- Bit 18 **SBKF**: Send break flag  
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.  
0: Break character transmitted  
1: Break character requested by setting SBKRQ bit in USART\_RQR register
- Bit 17 **CMF**: Character match flag  
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART\_ICR register.  
An interrupt is generated if CMIE = 1 in the USART\_CR1 register.  
0: No Character match detected  
1: Character Match detected
- Bit 16 **BUSY**: Busy flag  
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).  
0: USART is idle (no reception)  
1: Reception on going

**Bit 15 ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case). It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

**Bit 14 ABRE:** Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed).

It is cleared by software, by writing 1 to the ABRRQ bit in the USART\_RQR register.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.*

**Bit 13 UDR:** SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART\_TDR. This flag is reset by setting UDRCF bit in the USART\_ICR register.

0: No underrun error

1: underrun error

*Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 12 EOBF:** End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBI = 1 in the USART\_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART\_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

*Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 11 RTOF:** Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART\_ICR register.

An interrupt is generated if RTOIE = 1 in the USART\_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

*Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.*

*The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.*

*If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.*

**Bit 10 CTS:** CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

**Bit 9 CTSIF:** CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART\_ICR register.

An interrupt is generated if CTSIE = 1 in the USART\_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

**Bit 8 LBDF:** LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART\_ICR.

An interrupt is generated if LBDIE = 1 in the USART\_CR2 register.

0: LIN Break not detected

1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 7 TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART\_TDR. Every write operation to the USART\_TDR places the data in the TXFIFO.

This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART\_TDR.

An interrupt is generated if the TXFNFIE bit =1 in the USART\_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

*Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).*

*This bit is used during single buffer transmission.*

**Bit 6 TC:** Transmission complete

This bit indicates that the last data written in the USART\_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

An interrupt is generated if TCIE = 1 in the USART\_CR1 register.

TC bit is 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 33.5.8: Tolerance of the USART receiver to clock deviation on page 1004](#)).*

*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 FE CF 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 PE CF 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.*

**33.8.10 USART interrupt and status register [alternate] (USART\_ISR)**

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

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 TCBGTCTF in the USART\_ICR register or by a write to the USART\_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

*Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is '1'. Refer to [Section 33.4: USART implementation on page 986](#).*

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART\_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART\_ICR register. An interrupt is generated if WUFIE = 1 in the USART\_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*

*If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART\_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART\_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

- 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 33.4: USART implementation on page 986](#).*
- Bit 12 **EOBF**: End of block flag  
This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.  
An interrupt is generated if the EOBI = 1 in the USART\_CR1 register.  
It is cleared by software, writing 1 to the EOBCF in the USART\_ICR register.  
0: End of Block not reached  
1: End of Block (number of characters) reached  
*Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**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 LBD CF in the USART\_ICR.

An interrupt is generated if LBDIE = 1 in the USART\_CR2 register.

0: LIN Break not detected

1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*

**Bit 7 TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART\_TDR register has been transferred into the shift register. It is cleared by writing to the USART\_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART\_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART\_CR1 register.

0: Data register full

1: Data register not full

**Bit 6 TC:** Transmission complete

This bit indicates that the last data written in the USART\_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART\_CR1 register.

TC bit is 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 33.5.8: Tolerance of the USART receiver to clock deviation on page 1004](#)).*

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

### 33.8.11 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.
											w			w	
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 33.4: USART implementation on page 986](#).*

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 33.4: USART implementation on page 986](#)*

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 33.4: USART implementation on page 986](#).*

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 33.4: USART implementation on page 986](#).*

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 33.4: USART implementation on page 986](#).*

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 33.4: USART implementation on page 986](#).*

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.

### 33.8.12 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.	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.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

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

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 314](#)).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 33.8.13 USART transmit data register (USART\_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART\_TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 314](#)).

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.*

### 33.8.14 USART prescaler register (USART\_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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.	PRESCALER[3:0]			
												rw	rw	rw	rw

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.

*If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation on page 986](#).*



### 33.8.15 USART register map

The table below gives the USART register map and reset values.

**Table 193. 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	4	3	2	1	0
0x00	USART_CR1 FIFO enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]				DEDT[4:0]				OVER8	CMIE	MME	MO	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00	USART_CR1 FIFO disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]				DEDT[4:0]				OVER8	CMIE	MME	MO	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE		
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	USART_CR2	ADD[7:0]							RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP[1:0]	CLKEN	CPOL	CPHA	LBCL	Res.	LBIDIE	LBIDL	ADDM7	DIS_NSS	Res.	SIVEN			
	Reset value	0	0	0	0	0	0	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	TXFTCFG[2:0]		RXFTE	RXFTCFG[2:0]		TCBGTE	TXFTE	WUFIE	WUS[1:0]	SCAR CNT[2:0]		Res.	DEP	DEM	DDRE	OVRRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSSEL	IRLP	IREN	EIE				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]					PSC[7:0]										
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x14	USART_RTOR	BLEN[7:0]							RTO[23:0]																								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					X	X	X	X	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value																																
0x24	USART_RDR	RDR[8:0]																															
	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 193. 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	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDR[8:0]													
	Reset value																									0	0	0	0	0	0	0	0	0	0			
0x2C	USART_PRESC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PRESCALE R[3:0]			
	Reset value																																			0	0	0

Refer to [Section 2.2: Memory organization](#) for the register boundary addresses.

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

This section describes the low-power universal asynchronous receiver transmitter (LPUART).

### 34.1 LPUART introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the device is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

## 34.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 baud to 9600 baud using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data  
Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection

### 34.3 LPUART implementation

The table(s) below describe(s) LPUART implementation. It(they) also include(s) USARTs for comparison.

**Table 194. Instance implementation on STM32G0x1**

USART / LPUART instances	STM32G0x31xx, STM32G0x41xx, STM32G0x51xx, STM32G0x61xx	STM32G0x71xx, STM32G0x81xx	STM32G0xB1xx, STM32G0xC1xx
USART1	FULL	FULL	FULL
USART2	BASIC	FULL	FULL
USART3	-	BASIC	FULL
USART4	-	BASIC	BASIC
USART5	-	-	BASIC
USART6	-	-	BASIC
LPUART1	LP	LP	LP
LPUART2	-	-	LP

**Table 195. USART / LPUART features**

USART / LPUART modes/features <sup>(1)</sup>	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	X	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	-	-
LIN mode	X	-	-
Dual clock domain and wake-up from low-power mode	X	-	X
Receiver timeout interrupt	X	-	-
Modbus communication	X	-	-
Auto baud rate detection	X	-	-
Driver Enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	-	X
Tx/Rx FIFO size	8	-	8
Prescaler	X	-	X
Wake-up from low-power mode	X <sup>(2)</sup>	-	X <sup>(2)</sup>

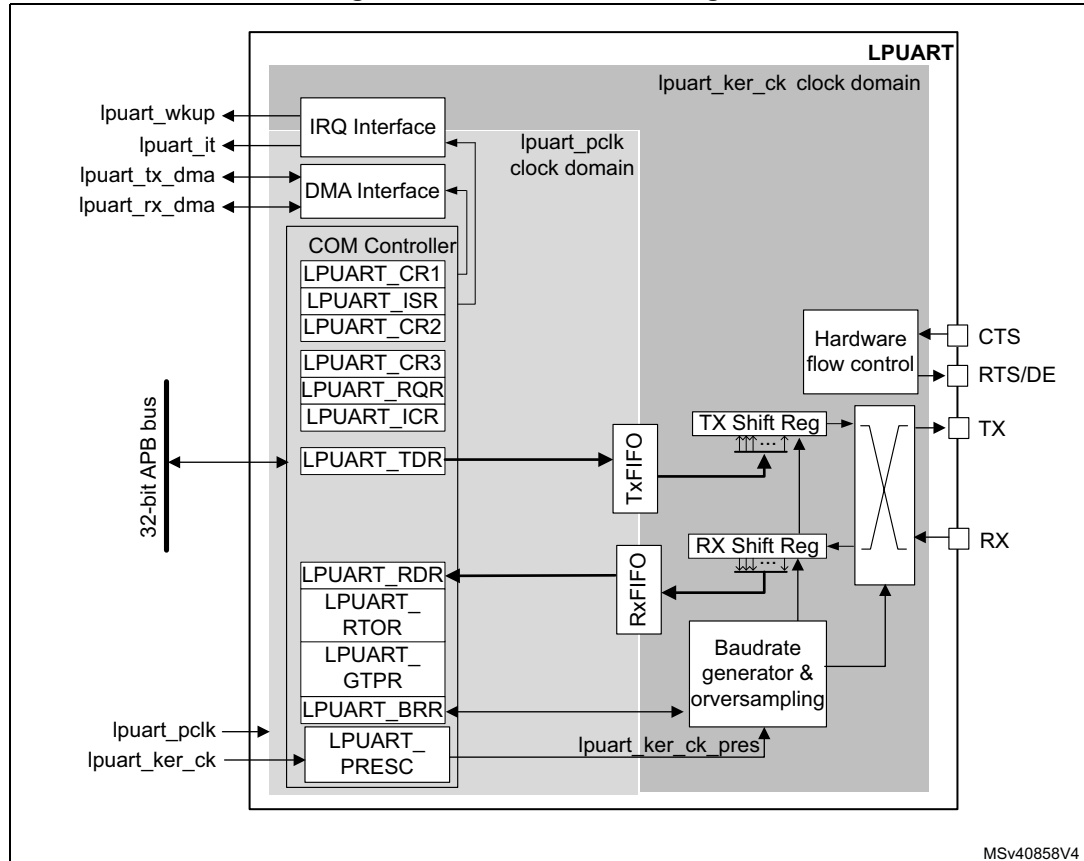
1. X = supported.

2. Wake-up supported from Stop 0 and Stop 1 modes.

### 34.4 LPUART functional description

#### 34.4.1 LPUART block diagram

Figure 341. LPUART block diagram



The simplified block diagram given in [Figure 341](#) shows two fully independent clock domains:

- The **lpuart\_pclk** clock domain

The **lpuart\_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart\_ker\_ck** kernel clock domain

The **lpuart\_ker\_ck** is the LPUART clock source. It is independent of the **lpuart\_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart\_ker\_ck** is stopped.

When the dual clock domain feature is disabled, the **lpuart\_ker\_ck** is the same as the **lpuart\_pclk** clock.

There is no constraint between **lpuart\_pclk** and **lpuart\_ker\_ck**: **lpuart\_ker\_ck** can be faster or slower than **lpuart\_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

### 34.4.2 LPUART signals

LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)  
RX is the serial data input.
- **TX** (Transmit Data Output)  
When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

#### RS232 hardware flow control mode

The following pins are required in RS232 hardware flow control mode:

- **CTS** (Clear To Send)  
When driven high, this signal blocks the data transmission at the end of the current transfer.
- **RTS** (Request to send)  
When it is low, this signal indicates that the USART is ready to receive data.

#### RS485 hardware flow control mode

The following pin is required in RS485 hardware control mode:

- **DE** (Driver Enable)  
This signal activates the transmission mode of the external transceiver.

*Note:* DE and RTS share the same pin.

Refer to [Table 196](#) and [Table 197](#) for the list of USART input/output pins and internal signals.

**Table 196. LPUART input/output pins**

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input
LPUART_TX	Output	Transmit data output
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE <sup>(1)</sup>	Output	Driver enable

1. LPUART\_DE and LPUART\_RTS share the same pin.

#### Description of USART input/output signals

**Table 197. LPUART internal input/output signals**

Pin name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	USART kernel clock

Table 197. LPUART internal input/output signals (continued)

Pin name	Signal type	Description
lpuart_wkup	Output	USART provides a wake-up interrupt
lpuart_it	Output	USART global interrupt
lpuart_tx_dma	Input/output	USART transmit DMA request
lpuart_rx_dma	Input/output	USART receive DMA request

### 34.4.3 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART\_CR1 register (see [Figure 315](#)).

- 7-bit character length: M[1:0] = '10'
- 8-bit character length: M[1:0] = '00'
- 9-bit character length: M[1:0] = '01'

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

An **Idle character** is interpreted as an entire frame of "1"s. (The number of "1" 's includes the number of stop bits).

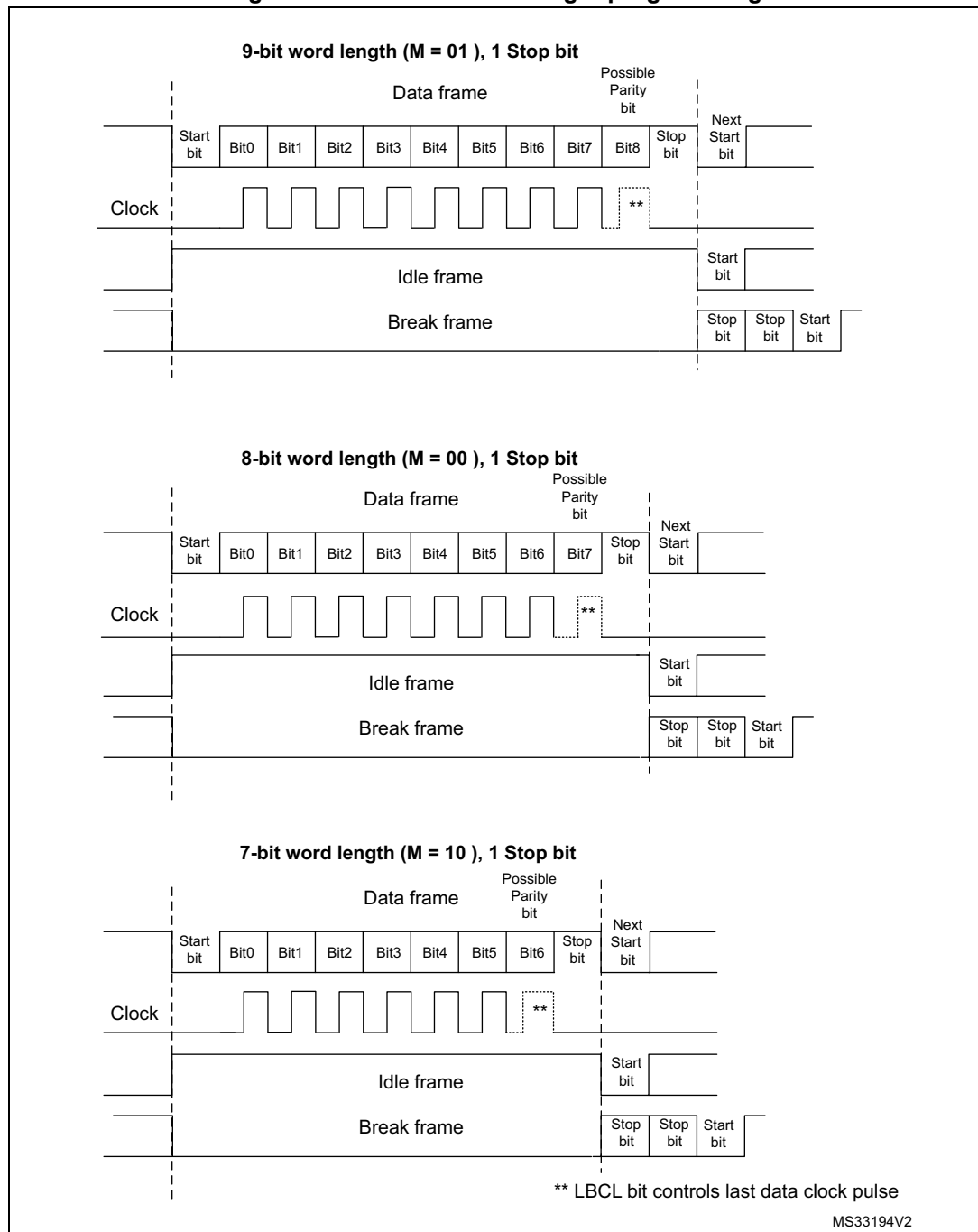
A **Break character** is interpreted on receiving "0"s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.



Figure 342. LPUART word length programming



### 34.4.4 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART\_CR1 register.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store



the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

*Note:* The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART\_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART\_CR3 control register.

In this case:

- The RXFT flag is set in the LPUART\_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.  
This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.  
RXFTCFG data have been received: one data in LPUART\_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to '101', the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART\_RDR. As a result, the next received data does not set the overrun flag.
- The TXFT flag is set in the LPUART\_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.  
This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

### 34.4.5 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

#### Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART\_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see [Figure 341](#)).

When FIFO mode is enabled, the data written to the LPUART\_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

*Note:* The TE bit must be set before writing the data to be transmitted to the LPUART\_TDR.

The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

### Configurable stop bits

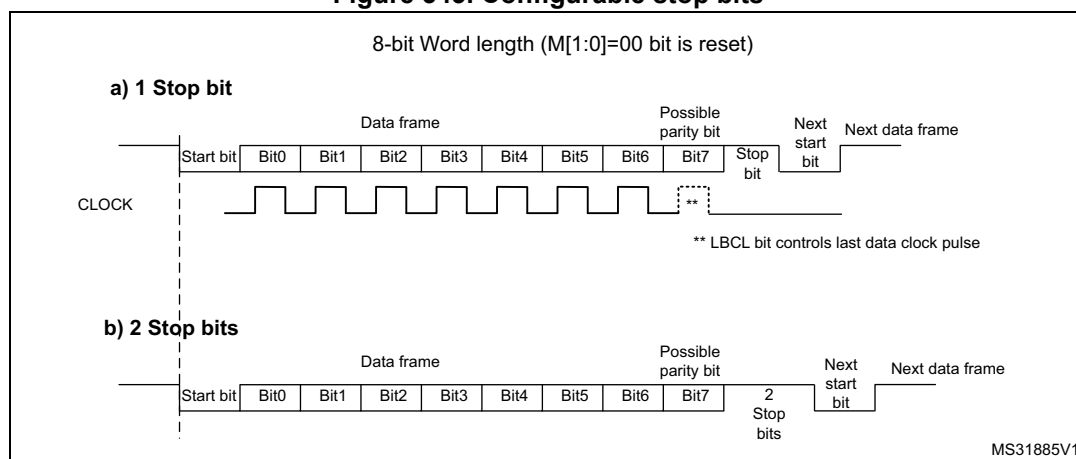
The number of stop bits to be transmitted with every character can be programmed in LPUART\_CR2 (bits 13,12).

- **1 stop bit:** This is the default value of number of stop bits.
- **2 Stop bits:** This is supported by normal LPUART, single-wire and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 343. Configurable stop bits**



### Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART\_CR1 to define the word length.
2. Select the desired baud rate using the LPUART\_BRR register.
3. Program the number of stop bits in LPUART\_CR2.
4. Enable the LPUART by writing the UE bit in LPUART\_CR1 register to '1'.
5. Select DMA enable (DMAT) in LPUART\_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in [Section 34.4.12: Continuous communication using DMA and LPUART](#).
6. Set the TE bit in LPUART\_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART\_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
  - When FIFO mode is disabled, writing a data in the LPUART\_TDR clears the TXE flag.
  - When FIFO mode is enabled, writing a data in the LPUART\_TDR adds one data to the TXFIFO. Write operations to the LPUART\_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART\_TDR register, wait until TC = 1. This indicates that the transmission of the last frame is complete.
  - When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.

- When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

### Single byte communication

- When FIFO mode disabled:

Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:

- the data have been moved from the LPUART\_TDR register to the shift register and data transmission has started;
- the LPUART\_TDR register is empty;
- the next data can be written to the LPUART\_TDR register without overwriting the previous data.

The TXE flag generates an interrupt if the TXEIE bit is set.

When a transmission is ongoing, a write instruction to the LPUART\_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

When no transmission is ongoing, a write instruction to the LPUART\_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware to indicate that:

- the TXFIFO is not full;
- the LPUART\_TDR register is empty;
- the next data can be written to the LPUART\_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the LPUART\_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

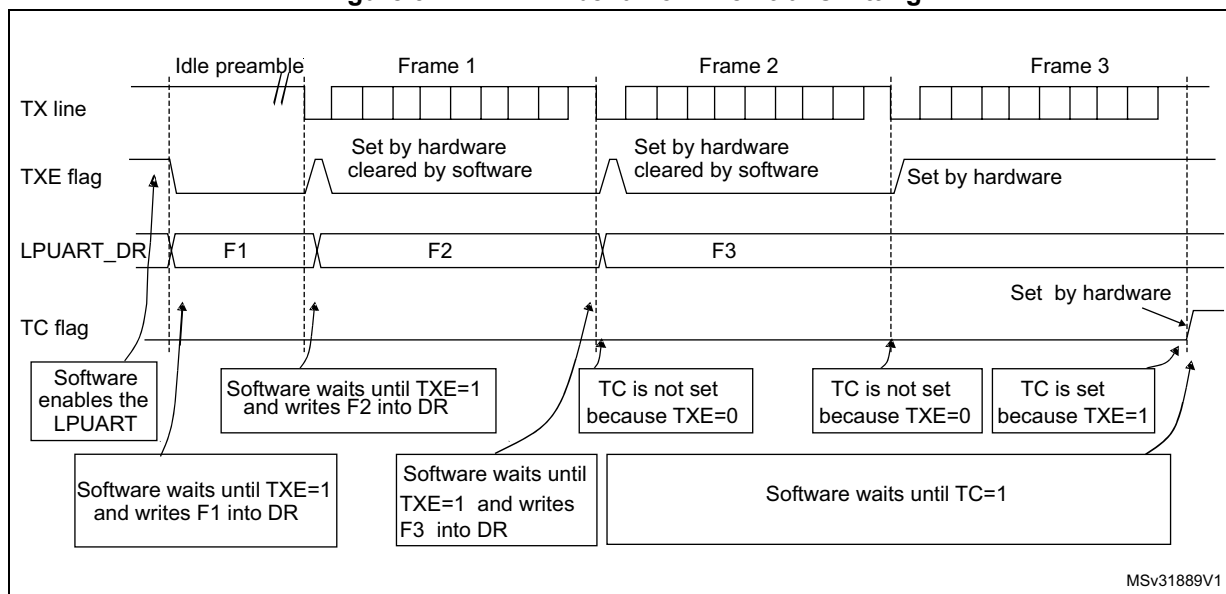
When the TXFIFO is not full, the TXFNF flag stays at '1' even after a write in LPUART\_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART\_CR1 register.

After writing the last data in the LPUART\_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the device to enter the low-power mode (see [Figure 344: TC/TXE behavior when transmitting](#)).

Figure 344. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

**Break characters**

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see Figure 342).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

**Idle characters**

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

**34.4.6 LPUART receiver**

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART\_CR1 register.

**Start bit detection**

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still '0'. If the start sample is at '1', then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

## Character reception

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART\_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

### Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART\_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART\_BRR.
3. Program the number of stop bits in LPUART\_CR2.
4. Enable the LPUART by writing the UE bit in LPUART\_CR1 register to '1'.
5. Select DMA enable (DMAR) in LPUART\_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 34.4.12: Continuous communication using DMA and LPUART](#).
6. Set the RE bit LPUART\_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART\_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty i.e. there is a data in the RXFIFO to be read.
- In single buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART\_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART\_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

### Break character

When a break character is received, the LPUART handles it as a framing error.

### Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

### Overrun error

- FIFO mode disabled
 

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available when a read to LPUART\_RDR is performed.;
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE bit or EIE bit is set.
- FIFO mode enabled
 

An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

Data can not be transferred from the shift register to the LPUART\_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  - the ORE bit is set;
  - the first entry in the RXFIFO is not lost. It is available when a read to LPUART\_RDR is performed.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

*Note: The ORE bit, when set, indicates that at least 1 data has been lost. T*

*When the FIFO mode is disabled, there are two possibilities*

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

### Selecting the clock source

The choice of the clock source is done through the Clock Control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

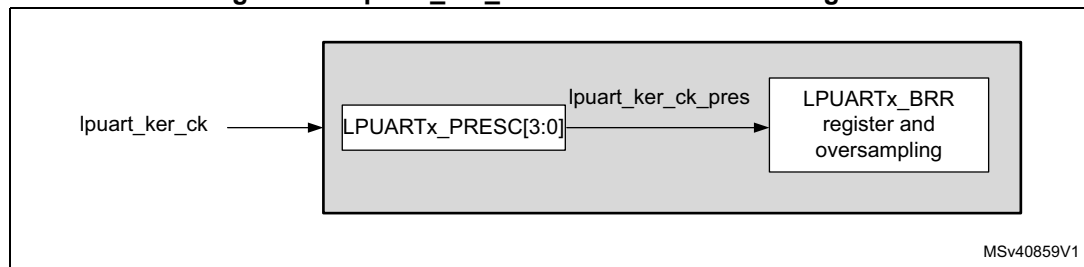
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is `lpuart_ker_ck`.

When the dual clock domain and the wake-up from low-power mode features are supported, the `lpuart_ker_ck` clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the `lpuart_ker_ck` is the same as `lpuart_pclk`.

The `lpuart_ker_ck` can be divided by a programmable factor in the LPUART\_PRESC register.

**Figure 345. lpuart\_ker\_ck clock divider block diagram**



Some `lpuart_ker_ck` sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART\_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

*Note:* There is no noise detection for data.

### Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART\_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of



multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART\_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART\_ICR register.

### Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART\_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits:** sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

### 34.4.7 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART\_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{Lpuartckpres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART\_BRR register.

*Note:* The baud counters are updated to the new value in the baud registers after a write operation to LPUART\_BRR. Hence the baud rate register value should not be changed during communication.

*It is forbidden to write values lower than 0x300 in the LPUART\_BRR register.*

*f<sub>CK</sub> must range from 3 x baud rate to 4096 x baud rate.*

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 baud. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbaud.

**Table 198. Error calculation for programmed baud rates at lpuart\_ker\_ck\_pres = 32.768 kHz**

Baud rate		lpuart_ker_ck_pres = 32.768 kHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	300 bauds	300 baud	0x6D3A	0
2	600 baud	600 baud	0x369D	0
3	1200 baud	1200.087 baud	0x1B4E	0.007
4	2400 baud	2400.17 baud	0xDA7	0.007
5	4800 baud	4801.72 baud	0x6D3	0.035
6	9600 baud	9608.94 baud	0x369	0.093

**Table 199. Error calculation for programmed baud rates at  $f_{CK} = 100 \text{ MHz}$**

Baud rate		$f_{CK} = 100\text{MHz}$		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate
1	38400 Baud	38400,04 Baud	A2C2A	0,0001
2	57600 Baud	57600,06 Baud	6C81C	0,0001
3	115200 Baud	115200,12 Baud	3640E	0,0001
4	230400 Baud	230400,23 Baud	1B207	0,0001
5	460800 Baud	460804,61 Baud	D903	0,001
6	921600 Baud	921625,81 Baud	6C81	0,0028
7	4000 Kbaud	4000000,00 Baud	1900	0
8	10000 Kbaud	10000000,00 Baud	A00	0
9	20000 Kbaud	20000000,00 Baud	500	0
10	33000 Kbaud	33032258,06 Baud	307	0,1

### 34.4.8 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in [Table 200](#):

- Number of Stop bits defined through STOP[1:0] bits in the LPUART\_CR2 register
- LPUART\_BRR register value.

Table 200. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = '00'), 1 Stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = '01'), 1 Stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = '10'), 1 Stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = '00'), 2 Stop bit	2.08%	2.86%	4.35%	4.42%
9 bits (M = '01'), 2 Stop bit	1.82%	2.56%	3.90%	4.42%
7 bits (M = '10'), 2 Stop bit	2.34%	3.23%	4.92%	4.42%

*Note:* The data specified in Table 200 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = '00' (11-bit times when M = '01' or 9-bit times when M = '10').

### 34.4.9 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART\_CR1 register.

*Note:* When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart\_ker\_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART\_ISR register is set to '1'. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART\_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART\_CR1 register:

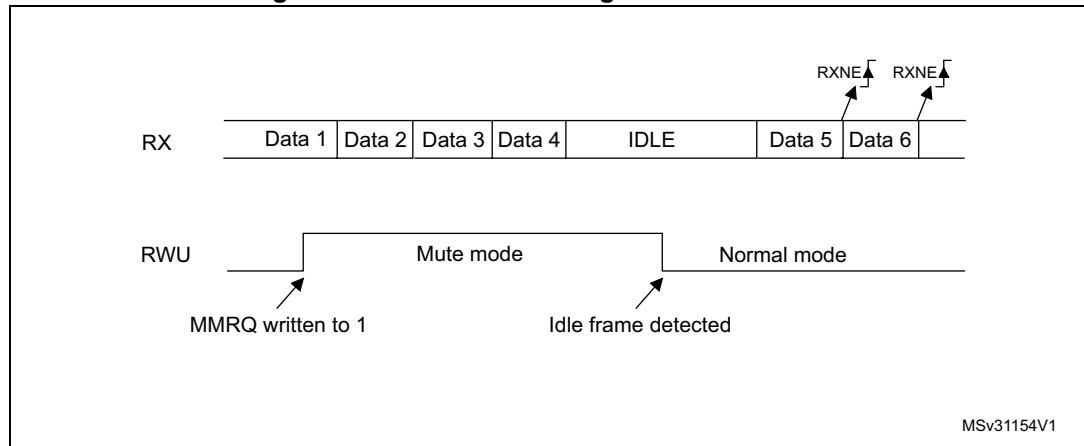
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

### Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART\_ISR register. An example of mute mode behavior using Idle line detection is given in [Figure 346](#).

**Figure 346. Mute mode using Idle line detection**



*Note:* If the MMRQ is set while the IDLE character has already elapsed, the mute mode is not entered (RWU is not set).  
 If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

### 4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a '1' otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART\_CR2 register.

*Note:* In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

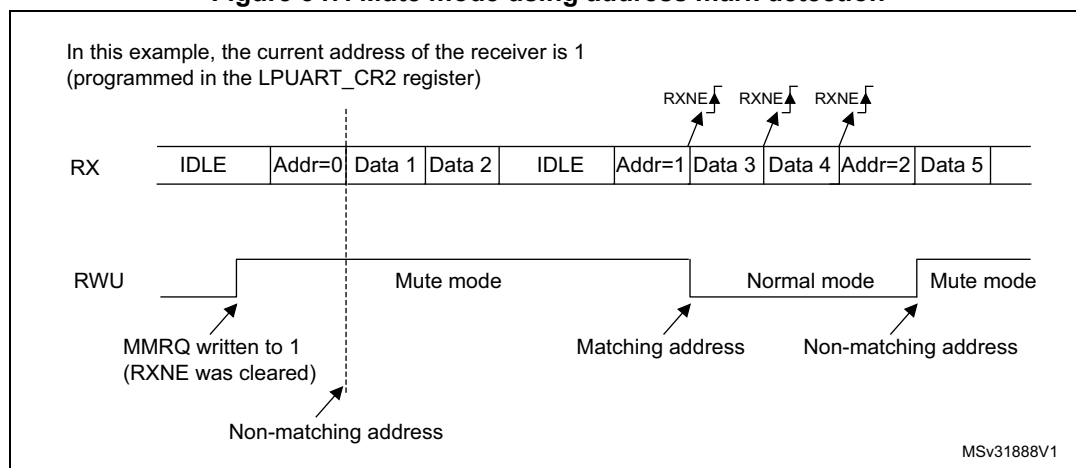
The LPUART also enters mute mode when the MMRQ bit is written to '1'. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

*Note:* When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in [Figure 347](#).

**Figure 347. Mute mode using address mark detection**



### 34.4.10 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART\_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in [Table 201](#).

**Table 201: LPUART frame formats**

M bits	PCE bit	LPUART 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 which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data equal 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART\_CR1 = 0).

#### Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data equal 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART\_CR1 = 1).

### Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART\_ISR register and an interrupt is generated if PEIE is set in the LPUART\_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART\_ICR register.

### Parity generation in transmission

If the PCE bit is set in LPUART\_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

## 34.4.11 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART\_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the LPUART\_CR2 register,
- SCEN and IREN bits in the LPUART\_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART\_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

*Note:* In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

## 34.4.12 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

*Note:* Refer to [Section 33.4: USART implementation on page 986](#) to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in [Section 33.5.6](#). To perform continuous communication. When FIFO is disabled, you can clear the TXE/ RXNE flags in the LPUART\_ISR register.

### Transmission using DMA

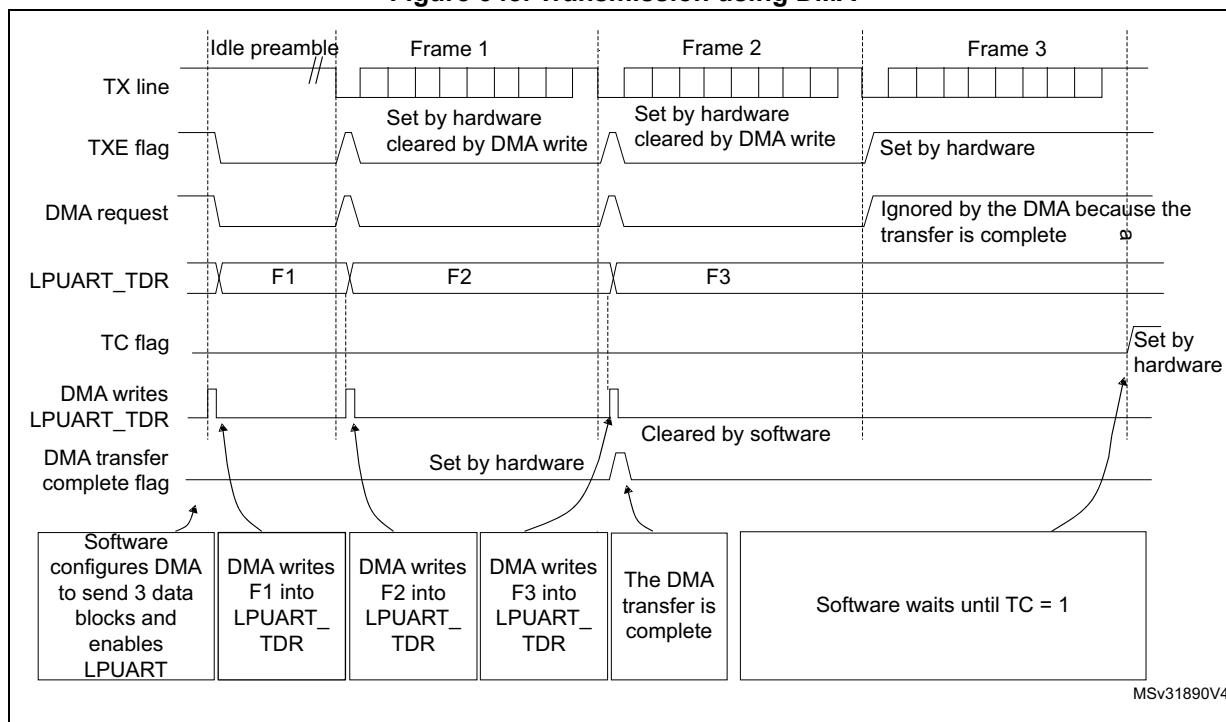
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART\_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding [Direct memory access controller section](#)) to the LPUART\_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

1. Write the LPUART\_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART\_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART\_ISR register by setting the TCCF bit in the LPUART\_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA\_ISR register), the TC flag can be monitored to make sure that the LPUART communication is complete. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 348. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

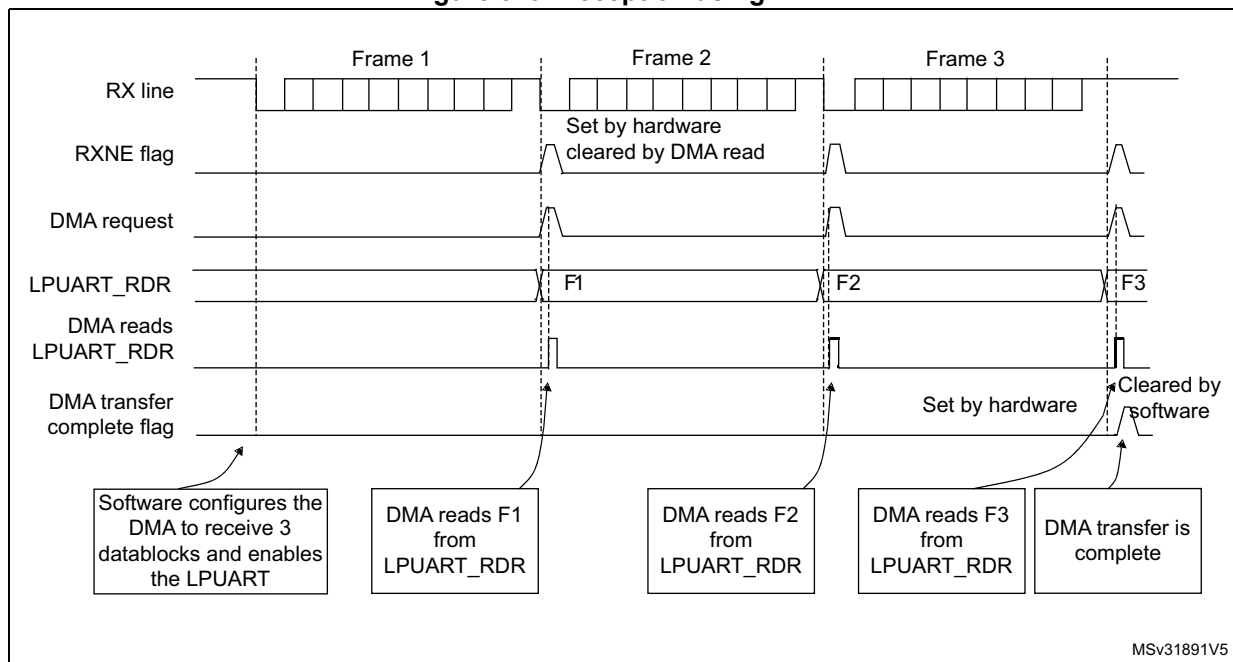
### Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART\_CR3 register. Data are loaded from the LPUART\_RDR register to a SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller (DMA)* section) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART\_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART\_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

**Figure 349. 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

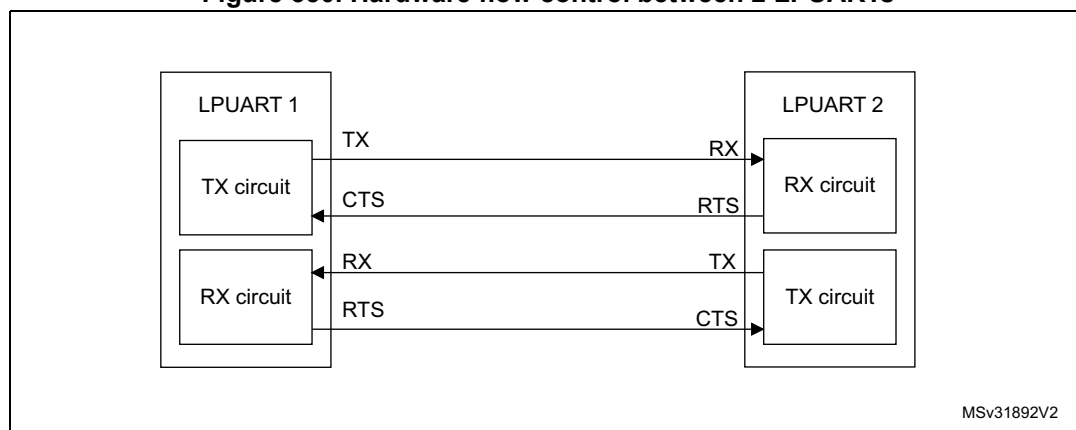


enable bit (EIE bit in the LPUART\_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

### 34.4.13 RS232 hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The [Figure 336](#) shows how to connect 2 devices in this mode:

**Figure 350. Hardware flow control between 2 LPUARTs**

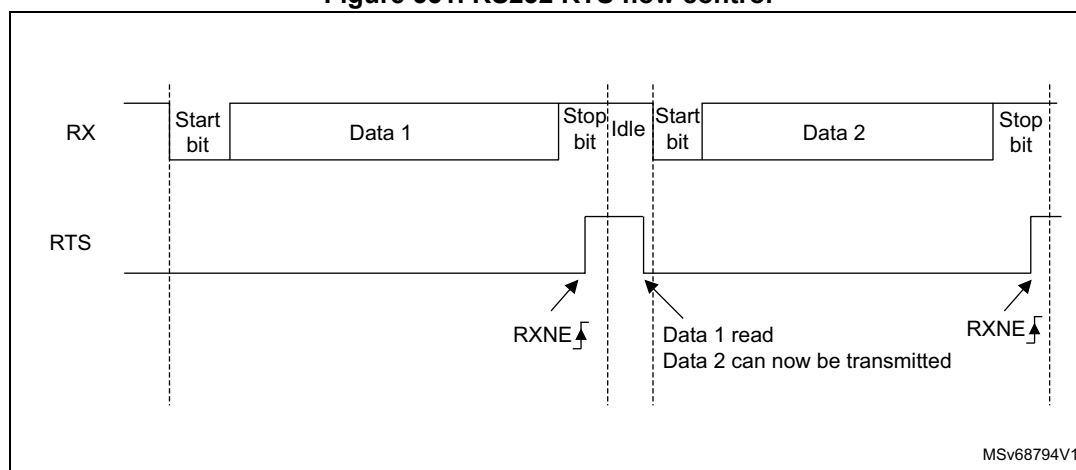


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART\_CR3 register).

#### RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. [Figure 351](#) shows an example of communication with RTS flow control enabled.

**Figure 351. RS232 RTS flow control**



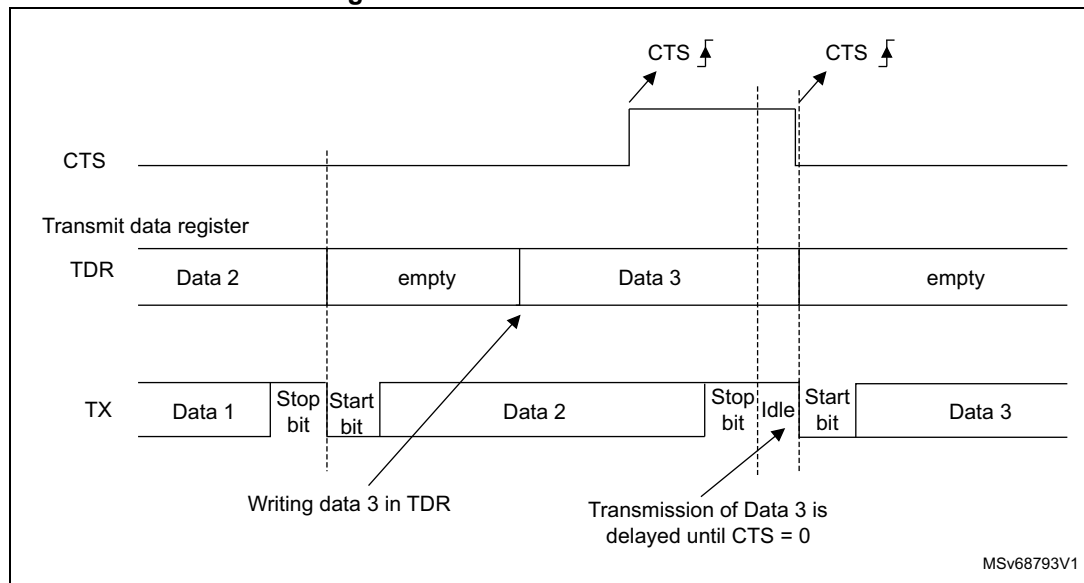
*Note:* When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

### RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART\_CR3 register is set. [Figure 352](#) shows an example of communication with CTS flow control enabled.

Figure 352. RS232 CTS flow control



**Note:** For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

### RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART\_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART\_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART\_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART\_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ( $f_{CK}$ ) cycles:

- The Driver enable assertion time equals
  - $(1 + (DEAT \times P)) \times f_{CK}$ , if  $P \neq 0$
  - $(1 + DEAT) \times f_{CK}$ , if  $P = 0$
- The Driver enable deassertion time equals
  - $(1 + (DEDT \times P)) \times f_{CK}$ , if  $P \neq 0$
  - $(1 + DEDT) \times f_{CK}$ , if  $P = 0$

where  $P = BRR[20:11]$

### 34.4.14 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the `lpuart_pclk` clock is disabled.

The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set. When the `lpuart_pclk` is gated, the LPUART provides a wake-up interrupt (`lpuart_wkup`) if a specific action requiring the activation of the `lpuart_pclk` clock is needed:

- If FIFO mode is disabled
  - `lpuart_pclk` clock has to be activated to empty the LPUART data register.
  - In this case, the `lpuart_wkup` interrupt source is the RXNE set to '1'. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled
  - `lpuart_pclk` clock has to be activated
    - to fill the TXFIFO
    - or to empty the RXFIFO
  - In this case, the `lpuart_wkup` interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set .
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

This 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 `lpuart_wkup` interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific `lpuart_wkup` interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and `lpuart_wkup` interrupt is generated if the WUFIE bit is set.

*Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.*

*The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.*

*When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.*

*When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.*

*When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.*

### Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

*Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (i.e. the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).*

### Wake-up from low-power mode when LPUART kernel clock `lpuart_ker_ck` is OFF in low-power mode

If during low-power mode, the `lpuart_ker_ck` clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the `lpuart_ker_ck` clock to be switched ON thanks to the `lpuart_ker_ck_req` signal. The `lpuart_ker_ck` is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the `lpuart_ker_ck` is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wake-up event programmed to “address match detection” and FIFO management disabled.

[Figure 353](#) shows the behavior when the wake-up event is verified.

**Figure 353. Wake-up event verified (wake-up event = address match, FIFO disabled)**

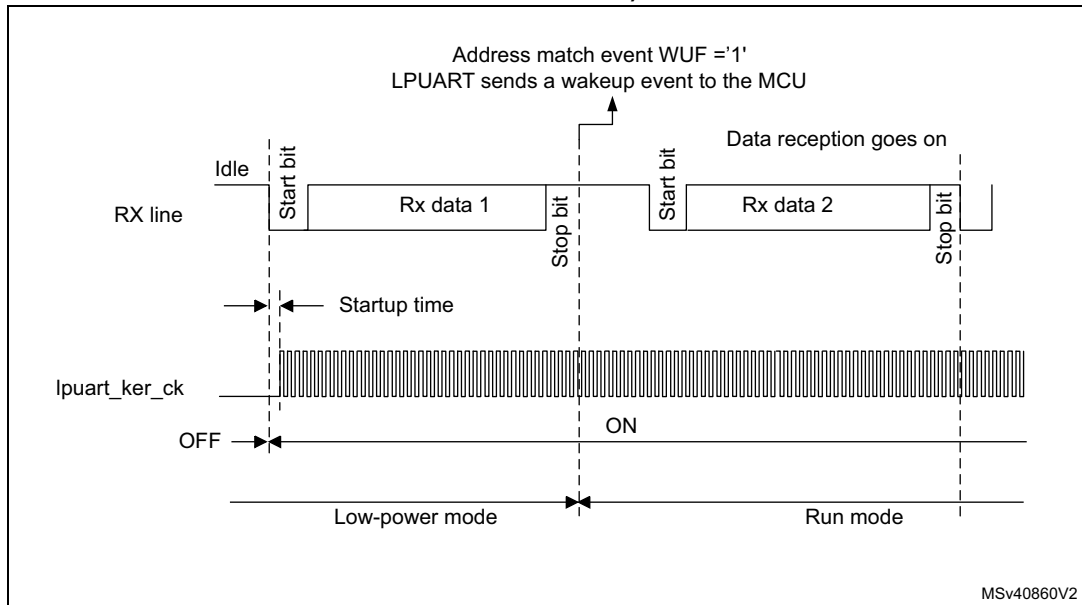
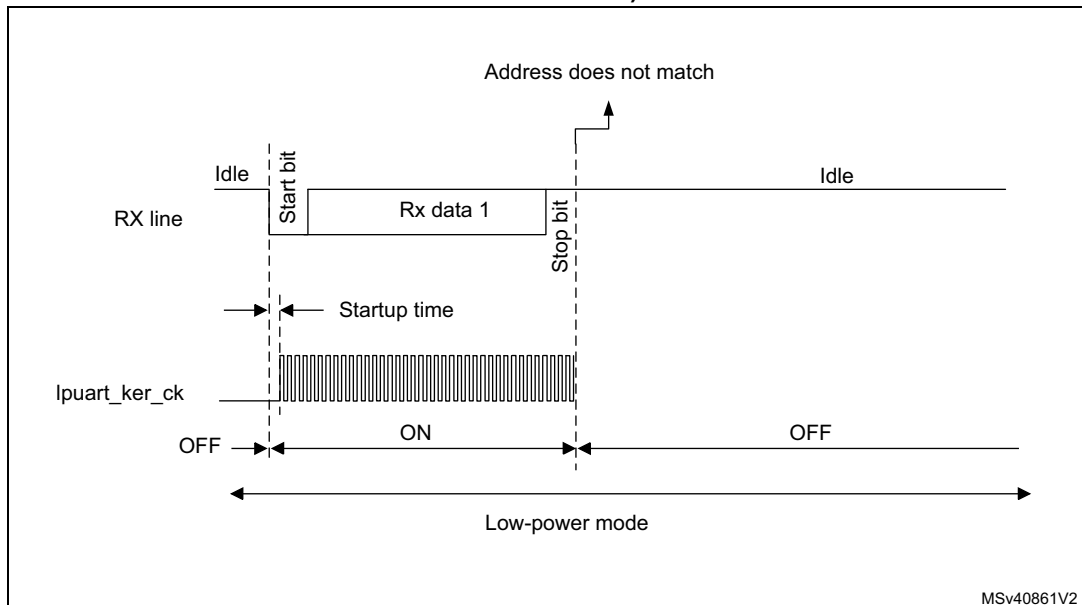


Figure 354 shows the behavior when the wake-up event is not verified.

**Figure 354. Wake-up event not verified (wake-up event = address match, FIFO disabled)**



*Note:* The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

### Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see [Section 34.4.8: Tolerance of the LPUART receiver to clock deviation](#)).

Let us take the example of OVER8 = 0, M bits = '01', ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to [Table 200: Tolerance of the LPUART receiver](#), the LPUART receiver tolerance equals 3.41%.

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

$$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$$

$$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$$

where  $t_{WULPUART}$  is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart\_ker\_ck inaccuracy.

For example, if HSI is used as lpuart\_ker\_ck, and the HSI inaccuracy is of 1%, then we obtain:

$$t_{WULPUART} = 3 \mu\text{s} \text{ (values provided only as examples; for correct values, refer to the device datasheet).}$$

$$D_{WUmax} = 3.41\% - 1\% = 2.41\%$$

$$T_{bit\ min} = 3 \mu\text{s} / (11 \times 2.41\%) = 11.32 \mu\text{s}.$$

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is:  $1/11.32 \mu\text{s} = 88.36 \text{ kbaud}$ .

## 34.5 LPUART in low-power modes

Table 202. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop <sup>(1)</sup>	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to [Section 34.3: LPUART implementation](#) to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

### 34.6 LPUART interrupts

Refer to [Table 203](#) for a detailed description of all LPUART interrupt requests.

**Table 203. LPUART 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
LPUART	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	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	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	RX-NEIE/RX-FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE		Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE <sup>(3)</sup>	EIE	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	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to [Section 34.3: LPUART implementation](#) for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART\_RDR. In Stop mode, LPUART\_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

### 34.7 LPUART registers

Refer to [Section 1.2 on page 55](#) for a list of abbreviations used in register descriptions.

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

#### 34.7.1 LPUART control register 1 (LPUART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

##### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXF FIE	TXFEIE	FIFO EN	M1	Res.	Res.	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
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFN FIE	TCIE	RXFN EIE	IDLEIE	TE	RE	UESM	UE
	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 is inhibited  
 1: An LPUART interrupt is generated when RXFF = 1 in the LPUART\_ISR register
- Bit 30 **TXFEIE**:TXFIFO empty interrupt enable  
 This bit is set and cleared by software.  
 0: Interrupt is inhibited  
 1: An LPUART interrupt is generated when TXFE = 1 in the LPUART\_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.





**Bit 28 M1:** Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE = 0).

*Note:* In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

**Bits 25:21 DEAT[4:0]:** Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart\_ker\_ck clock cycles. For more details, refer [Section 33.5.20: RS232 hardware flow control and RS485 Driver Enable](#).

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bits 20:16 DEDT[4:0]:** Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart\_ker\_ck clock cycles. For more details, refer [Section 34.4.13: RS232 hardware flow control and RS485 Driver Enable](#).

If the LPUART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

**Bit 14 CMIE:** Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART\_ISR register.

**Bit 13 MME:** Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

**Bit 12 M0:** Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

**Bit 11 WAKE:** Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bit 10 PCE:** Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bit 9 PS:** Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bit 8 PEIE:** PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART\_ISR register

**Bit 7 TXFNFIE:** TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF = 1 in the LPUART\_ISR register

**Bit 6 TCIE:** Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART\_ISR register

**Bit 5 RXFNEIE:** RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE/RXFNE = 1 in the LPUART\_ISR register

**Bit 4 IDLEIE:** IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART\_ISR register

**Bit 3 TE:** Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART\_ISR register.*

*When TE is set there is a 1 bit-time delay before the transmission starts.*

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode.

When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.*

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART\_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

### 34.7.2 LPUART control register 1 [alternate] (LPUART\_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

#### FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		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.	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

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

**Bit 28 M1:** Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 Start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 Start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 Start bit, 7 Data bits, n Stop bit

This bit can only be written when the LPUART is disabled (UE = 0).

*Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bits 27:26 Reserved, must be kept at reset value.

**Bits 25:21 DEAT[4:0]:** Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart\_ker\_ck clock cycles. For more details, refer [Section 33.5.20: RS232 hardware flow control and RS485 Driver Enable](#).

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bits 20:16 DEDT[4:0]:** Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart\_ker\_ck clock cycles. For more details, refer [Section 34.4.13: RS232 hardware flow control and RS485 Driver Enable](#).

If the LPUART\_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

**Bit 14 CMIE:** Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART\_ISR register.

**Bit 13 MME:** Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

**Bit 12 M0:** Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

**Bit 11 WAKE:** Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bit 10 PCE:** Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bit 9 PS:** Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

**Bit 8 PEIE:** PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART\_ISR register

**Bit 7 TXEIE:** Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE/TXFNF = 1 in the LPUART\_ISR register

**Bit 6 TCIE:** Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART\_ISR register

**Bit 5 RXNEIE:** Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE/RXFNE = 1 in the LPUART\_ISR register

**Bit 4 IDLEIE:** IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART\_ISR register

**Bit 3 TE:** Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

*Note: During transmission, a low pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word. In order to generate an idle character, the TE must not be immediately written to 1. In order to ensure the required duration, the software can poll the TEACK bit in the LPUART\_ISR register.*

*When TE is set there is a 1 bit-time delay before the transmission starts.*

**Bit 2 RE:** Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in Stop mode

When this bit is cleared, the LPUART is not able to wake up the MCU from low-power mode. When this bit is set, the LPUART is able to wake up the MCU from low-power mode, provided that the LPUART clock selection is HSI or LSE in the RCC.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode. When this function is active, the clock source for the LPUART must be HSI or LSE (see RCC chapter)

*Note: It is recommended to set the UESM bit just before entering low-power mode and clear it on exit from low-power mode.*

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART\_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

*Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART\_ISR to be set before resetting the UE bit.*

*The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.*

### 34.7.3 LPUART control register 2 (LPUART\_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]								Res.	Res.	Res.	Res.	MSBFI RST	DATAIN V	TXINV	RXINV
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	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV**: Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV**: TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , 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 LPUART is disabled (UE = 0).

Bit 16 **RXINV**: RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels ( $V_{DD} = 1/\text{idle}$ , 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 LPUART is disabled (UE = 0).

Bit 15 **SWAP**: Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: STOP bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: Reserved.

10: 2 stop bits

11: Reserved

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the LPUART is disabled (UE = 0)

*Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.*

Bits 3:0 Reserved, must be kept at reset value.

### 34.7.4 LPUART control register 3 (LPUART\_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]			RXFTIE			RXFTCFG[2:0]			Res.			TXFTIE			WUFIE			WUS[1:0]			Res.			Res.			Res.			Res.																	
rw			rw			rw			rw			rw			rw			rw			rw			rw			rw			rw			rw														
15			14			13			12			11			10			9			8			7			6			5			4			3			2			1			0		
DEP			DEM			DDRE			OVRDIS			Res.			CTSIE			CTSE			RTSE			DMAT			DMAR			Res.			Res.			HDSEL			Res.			Res.			EIE		
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.

110: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 is inhibited

1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.



- Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration  
 000:Receive FIFO reaches 1/8 of its depth.  
 001:Receive FIFO reaches 1/4 of its depth.  
 110:Receive FIFO reaches 1/2 of its depth.  
 011:Receive FIFO reaches 3/4 of its depth.  
 100:Receive FIFO reaches 7/8 of its depth.  
 101:Receive FIFO becomes full.  
 Remaining combinations: Reserved.
- Bit 24 Reserved, must be kept at reset value.
- Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable  
 This bit is set and cleared by software.  
 0: Interrupt is inhibited  
 1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.
- Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable  
 This bit is set and cleared by software.  
 0: Interrupt is inhibited  
 1: An LPUART interrupt is generated whenever WUF = 1 in the LPUART\_ISR register  
*Note: WUFIE must be set before entering in low-power mode.*  
*If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation](#).*
- Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection  
 This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).  
 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)  
 01:Reserved.  
 10: WUF active on Start bit detection  
 11: WUF active on RXNE.  
 This bitfield can only be written when the LPUART is disabled (UE = 0).  
*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to [Section 33.4: USART implementation](#).*
- Bits 19:16 Reserved, must be kept at reset value.
- Bit 15 **DEP**: Driver enable polarity selection  
 0: DE signal is active high.  
 1: DE signal is active low.  
 This bit can only be written when the LPUART is disabled (UE = 0).
- Bit 14 **DEM**: Driver enable mode  
 This bit enables the user to activate the external transceiver control, through the DE signal.  
 0: DE function is disabled.  
 1: DE function is enabled. The DE signal is output on the RTS pin.  
 This bit can only be written when the LPUART is disabled (UE = 0).

- Bit 13 **DDRE**: DMA disable on reception error
- 0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
  - 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
- This bit can only be written when the LPUART is disabled (UE = 0).
- Note: The reception errors are: parity error, framing error or noise error.*
- Bit 12 **OVRDIS**: Overrun disable
- This bit is used to disable the receive overrun detection.
- 0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
  - 1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART\_RDR register.
- This bit can only be written when the LPUART is disabled (UE = 0).
- Note: This control bit enables checking the communication flow w/o reading the data.*
- Bit 11 Reserved, must be kept at reset value.
- Bit 10 **CTSIE**: CTS interrupt enable
- 0: Interrupt is inhibited
  - 1: An interrupt is generated whenever CTSIF = 1 in the LPUART\_ISR register
- Bit 9 **CTSE**: CTS enable
- 0: CTS hardware flow control disabled
  - 1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
- This bit can only be written when the LPUART is disabled (UE = 0)
- Bit 8 **RTSE**: RTS enable
- 0: RTS hardware flow control disabled
  - 1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
- This bit can only be written when the LPUART is disabled (UE = 0).
- Bit 7 **DMAT**: DMA enable transmitter
- This bit is set/reset by software
- 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
- Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART\_ISR register).

0: Interrupt is inhibited

1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART\_ISR register.

### 34.7.5 LPUART baud rate register (LPUART\_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **BRR[19:0]**: LPUART baud rate

*Note: It is forbidden to write values lower than 0x300 in the LPUART\_BRR register. Provided that LPUART\_BRR must be ≥ 0x300 and LPUART\_BRR is 20 bits, a care should be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].*

### 34.7.6 LPUART request register (LPUART\_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.	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.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART\_ISR register).

*Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.*

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

*Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.*

Bit 0 Reserved, must be kept at reset value.

### 34.7.7 LPUART interrupt and status register (LPUART\_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

#### FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					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 in LPUART\_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART\_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

- Bit 26 **RXFT**: RXFIFO threshold flag  
This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART\_CR3 register i.e. the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the LPUART\_CR3 register.  
0: Receive FIFO does not reach the programmed threshold.  
1: Receive FIFO reached the programmed threshold.
- Bit 25 Reserved, must be kept at reset value.
- Bit 24 **RXFF**: RXFIFO full  
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART\_RDR register).  
An interrupt is generated if the RXFFIE bit = 1 in the LPUART\_CR1 register.  
0: RXFIFO is not full  
1: RXFIFO is full
- Bit 23 **TXFE**: TXFIFO empty  
This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART\_RQR register.  
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART\_CR1 register.  
0: TXFIFO is not empty  
1: TXFIFO is empty
- Bit 22 **REACK**: Receive enable acknowledge flag  
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.  
It can be used to verify that the LPUART is ready for reception before entering low-power mode.  
*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.*
- Bit 21 **TEACK**: Transmit enable acknowledge flag  
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.  
It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART\_CR1 register, in order to respect the TE = 0 minimum period.
- Bit 20 **WUF**: Wake-up from low-power mode flag  
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART\_ICR register.  
An interrupt is generated if WUFIE = 1 in the LPUART\_CR3 register.  
*Note: When UESM is cleared, WUF flag is also cleared.*  
*If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value*
- Bit 19 **RWU**: Receiver wake-up from mute mode  
This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART\_CR1 register.  
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART\_RQR register.  
0: Receiver in Active mode  
1: Receiver in mute mode  
*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.*

- Bit 18 **SBKF**: Send break flag  
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.  
0: Break character transmitted  
1: Break character requested by setting SBKRQ bit in LPUART\_RQR register
- Bit 17 **CMF**: Character match flag  
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART\_ICR register.  
An interrupt is generated if CMIE = 1 in the LPUART\_CR1 register.  
0: No Character match detected  
1: Character Match detected
- Bit 16 **BUSY**: Busy flag  
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).  
0: LPUART is idle (no reception)  
1: Reception on going
- Bits 15:11 Reserved, must be kept at reset value.
- Bit 10 **CTS**: CTS flag  
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.  
0: CTS line set  
1: CTS line reset  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*
- Bit 9 **CTSIF**: CTS interrupt flag  
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART\_ICR register.  
An interrupt is generated if CTSIE = 1 in the LPUART\_CR3 register.  
0: No change occurred on the CTS status line  
1: A change occurred on the CTS status line  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*
- Bit 8 Reserved, must be kept at reset value.
- Bit 7 **TXFNF**: TXFIFO not full  
TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART\_TDR. Every write in the LPUART\_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART\_TDR.  
The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).  
An interrupt is generated if the TXFNFIE bit = 1 in the LPUART\_CR1 register.  
0: Data register is full/Transmit FIFO is full.  
1: Data register/Transmit FIFO is not full.  
*Note: This bit is used during single buffer transmission.*

**Bit 6 TC:** Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXFF is set. An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART\_ICR register or by a write to the LPUART\_TDR register.

An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.*

**Bit 5 RXFNE:** RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART\_RDR register. Every read of the LPUART\_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

**Bit 4 IDLE:** Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).*

*If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

**Bit 3 ORE:** Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART\_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART\_ICR register.

An interrupt is generated if RXFNEIE = 1 or EIE = 1 in the LPUART\_CR1 register, or EIE = 1 in the LPUART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the LPUART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART\_CR3 register.*

**Bit 2 NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART\_ICR register.

0: No noise is detected

1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

*This error is associated with the character in the LPUART\_RDR.*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART\_ICR register. When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART\_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

*Note: This error is associated with the character in the LPUART\_RDR.*

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART\_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART\_CR1 register.

0: No parity error

1: Parity error

*Note: This error is associated with the character in the LPUART\_RDR.*

### 34.7.8 LPUART interrupt and status register [alternate] (LPUART\_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled**

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.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART\_CR1 register, in order to respect the TE = 0 minimum period.



- Bit 20 **WUF**: Wake-up from low-power mode flag  
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART\_ICR register. An interrupt is generated if WUFIE = 1 in the LPUART\_CR3 register.  
*Note: When UESM is cleared, WUF flag is also cleared.  
If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value*
- Bit 19 **RWU**: Receiver wake-up from mute mode  
This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART\_CR1 register.  
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART\_RQR register.  
0: Receiver in active mode  
1: Receiver in mute mode  
*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.*
- Bit 18 **SBKF**: Send break flag  
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART\_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.  
0: Break character transmitted  
1: Break character requested by setting SBKRQ bit in LPUART\_RQR register
- Bit 17 **CMF**: Character match flag  
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART\_ICR register.  
An interrupt is generated if CMIE = 1 in the LPUART\_CR1 register.  
0: No Character match detected  
1: Character Match detected
- Bit 16 **BUSY**: Busy flag  
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).  
0: LPUART is idle (no reception)  
1: Reception on going
- Bits 15:11 Reserved, must be kept at reset value.
- Bit 10 **CTS**: CTS flag  
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.  
0: CTS line set  
1: CTS line reset  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*
- Bit 9 **CTSIF**: CTS interrupt flag  
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART\_ICR register.  
An interrupt is generated if CTSIE = 1 in the LPUART\_CR3 register.  
0: No change occurred on the CTS status line  
1: A change occurred on the CTS status line  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **Reserved**, must be kept at reset value.

Bit 7 **TXE**: Transmit data register empty/TXFIFO not full

TXE is set by hardware when the content of the LPUART\_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART\_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART\_CR1 register.

0: Data register full

1: Data register not full

*Note: This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the TCCF in the LPUART\_ICR register or by a write to the LPUART\_TDR register.

An interrupt is generated if TCIE = 1 in the LPUART\_CR1 register.

0: Transmission is not complete

1: Transmission is complete

*Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.*

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART\_RDR shift register has been transferred to the LPUART\_RDR register. It is cleared by reading from the LPUART\_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART\_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART\_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART\_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART\_ICR register.

0: No Idle line is detected

1: Idle line is detected

*Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs). If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.*

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART\_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART\_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART\_CR1 register, or EIE = 1 in the LPUART\_CR3 register.

0: No overrun error

1: Overrun error is detected

*Note: When this bit is set, the LPUART\_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

*This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART\_CR3 register.*

Bit 2 **NE**: Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NECF bit in the LPUART\_ICR register.

- 0: No noise is detected
- 1: Noise is detected

*Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.*

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART\_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame). An interrupt is generated if EIE = 1 in the LPUART\_CR3 register.

- 0: No Framing error is detected
- 1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the LPUART\_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART\_CR1 register.  
 0: No parity error  
 1: Parity error

### 34.7.9 LPUART interrupt flag clear register (LPUART\_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						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 LPUART\_ISR register.

*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 33.4: USART implementation.*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART\_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART\_ISR register.



Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag  
 Writing 1 to this bit clears the TC flag in the LPUART\_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag  
 Writing 1 to this bit clears the IDLE flag in the LPUART\_ISR register.

Bit 3 **ORECF**: Overrun error clear flag  
 Writing 1 to this bit clears the ORE flag in the LPUART\_ISR register.

Bit 2 **NECF**: Noise detected clear flag  
 Writing 1 to this bit clears the NE flag in the LPUART\_ISR register.

Bit 1 **FECF**: Framing error clear flag  
 Writing 1 to this bit clears the FE flag in the LPUART\_ISR register.

Bit 0 **PECF**: Parity error clear flag  
 Writing 1 to this bit clears the PE flag in the LPUART\_ISR register.

### 34.7.10 LPUART receive data register (LPUART\_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.	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.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

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

Bits 8:0 **RDR[8:0]**: Receive data value  
 Contains the received data character.  
 The RDR register provides the parallel interface between the input shift register and the internal bus (see [Figure 341](#)).  
 When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 34.7.11 LPUART transmit data register (LPUART\_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw



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

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see [Figure 341](#)).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART\_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXFNF = 1.*

### 34.7.12 LPUART prescaler register (LPUART\_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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.	PRESCALER[3:0]			
												rw	rw	rw	rw

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided

0001: input clock divided by 2

0010: input clock divided by 4

0011: input clock divided by 6

0100: input clock divided by 8

0101: input clock divided by 10

0110: input clock divided by 12

0111: input clock divided by 16

1000: input clock divided by 32

1001: input clock divided by 64

1010: input clock divided by 128

1011: input clock divided by 256

Remaining combinations: Reserved.

*Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.*

*If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to [Section 34.3: LPUART implementation on page 1073](#).*

### 34.7.13 LPUART register map

The table below gives the LPUART register map and reset values.

**Table 204. LPUART 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	LPUART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]				DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE			
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x00	LPUART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]				DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE			
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x04	LPUART_CR2	ADD[7:0]							Res.	Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]	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	
0x08	LPUART_CR3	TXFTCFG[2:0]		RXFTIE			RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS [1:0]	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	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	
0x0C	LPUART_BRR	BRR[19:0]																																
	Reset value																																	
0x10- 0x14	Reserved																																	
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		
	Reset value																																	
0x1C	LPUART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXENF	TC	RXNE	IDLE	TXFRQ	ORE	MMRO	SBKRO
	Reset value					0	0		0	1	0	0	0	0	0	0	0							0	0		1	1	0	0	0	0	0	0
0x1C	LPUART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0	0	0	0	0	0							0	0		1	1	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	
	Reset value												0			0																		
0x24	LPUART_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.	RDR[8:0]				Res.	Res.	Res.	Res.	
	Reset value																									0	0	0	0	0	0	0	0	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]				Res.	Res.	Res.	Res.		
	Reset value																								0	0	0	0	0	0	0	0	0	0



Table 204. LPUART 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			
0x2C	LPUART_PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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

Refer to [Section 2.2: Memory organization](#) for the register boundary addresses.

## 35 Serial peripheral interface / integrated interchip sound (SPI/I<sup>2</sup>S)

### 35.1 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.

### 35.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- 4 to 16-bit data size selection
- Multimaster mode capability
- 8 master mode baud rate prescalers up to  $f_{PCLK}/2$
- Slave mode frequency up to  $f_{PCLK}/2$
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI Motorola support
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun flags with interrupt capability
- CRC Error flag
- Two 32-bit embedded Rx and Tx FIFOs with DMA capability
- Enhanced TI and NSS pulse modes support



### 35.3 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).

### 35.4 SPI/I2S implementation

The following table describes all the SPI instances and their features embedded in the devices.

**Table 205. STM32G0x1 SPI and SPI/I2S implementation**

SPI Features	SPI1 / I2S1	SPI2 / I2S2 <sup>(1)</sup>	SPI3 <sup>(1)</sup>
Enhanced NSSP & TI modes	Yes	Yes	Yes
I2S support	Yes	Yes <sup>(1)</sup> / No	No
Hardware CRC calculation	Yes	Yes	Yes
Data size configuration	from 4 to 16 bits	from 4 to 16 bits	from 4 to 16 bits
Rx/Tx FIFO size	32 bits	32 bits	32 bits
Wake-up capability from Low-power Sleep	Yes	Yes	Yes

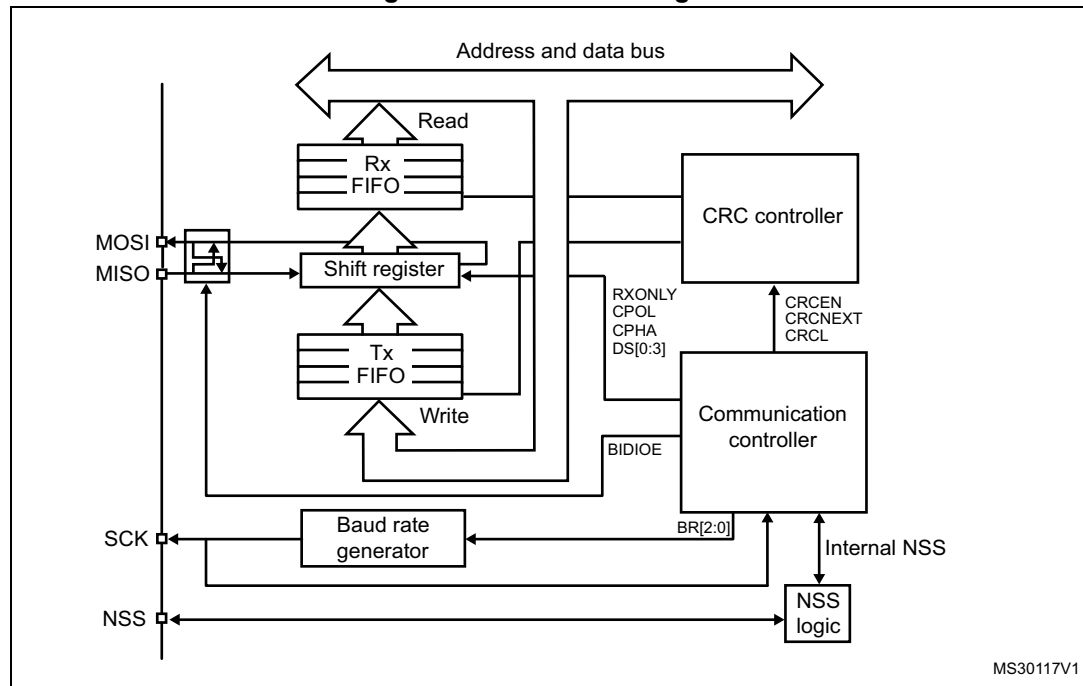
1. Applies to STM32G0B1xx and STM32G0C1xx only.

## 35.5 SPI functional description

### 35.5.1 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 355](#).

Figure 355. SPI block diagram



Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

See [Section 35.5.5: 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.

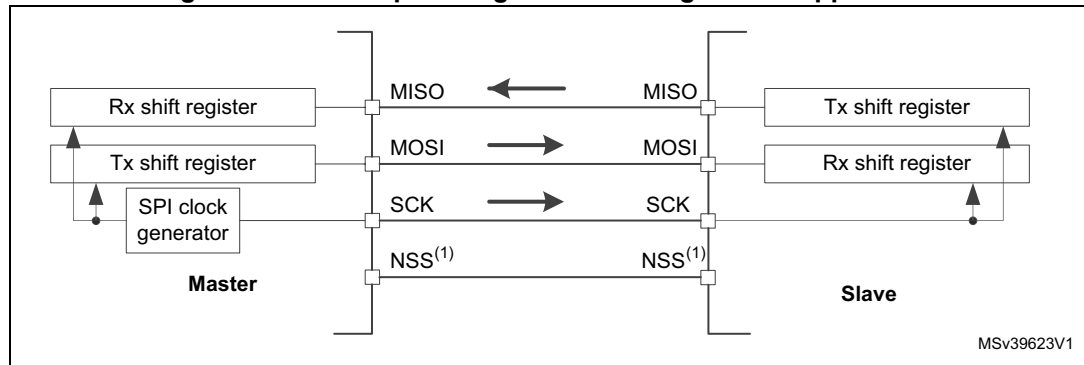
### 35.5.2 Communications between one master and one slave

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 356. Full-duplex single master/ single slave application

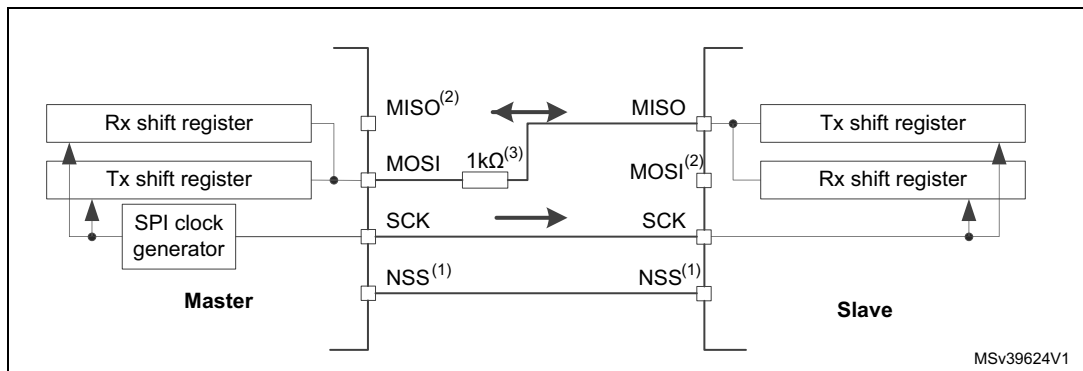


1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.5.5: Slave select \(NSS\) pin management](#).

#### 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 357. Half-duplex single master/ single slave application



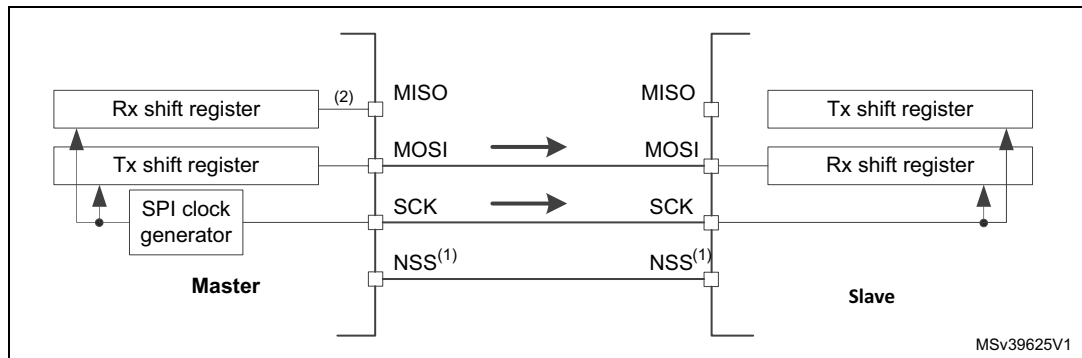
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.5.5: Slave select \(NSS\) pin management](#).
2. In 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 [35.5.5: 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 358. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)**



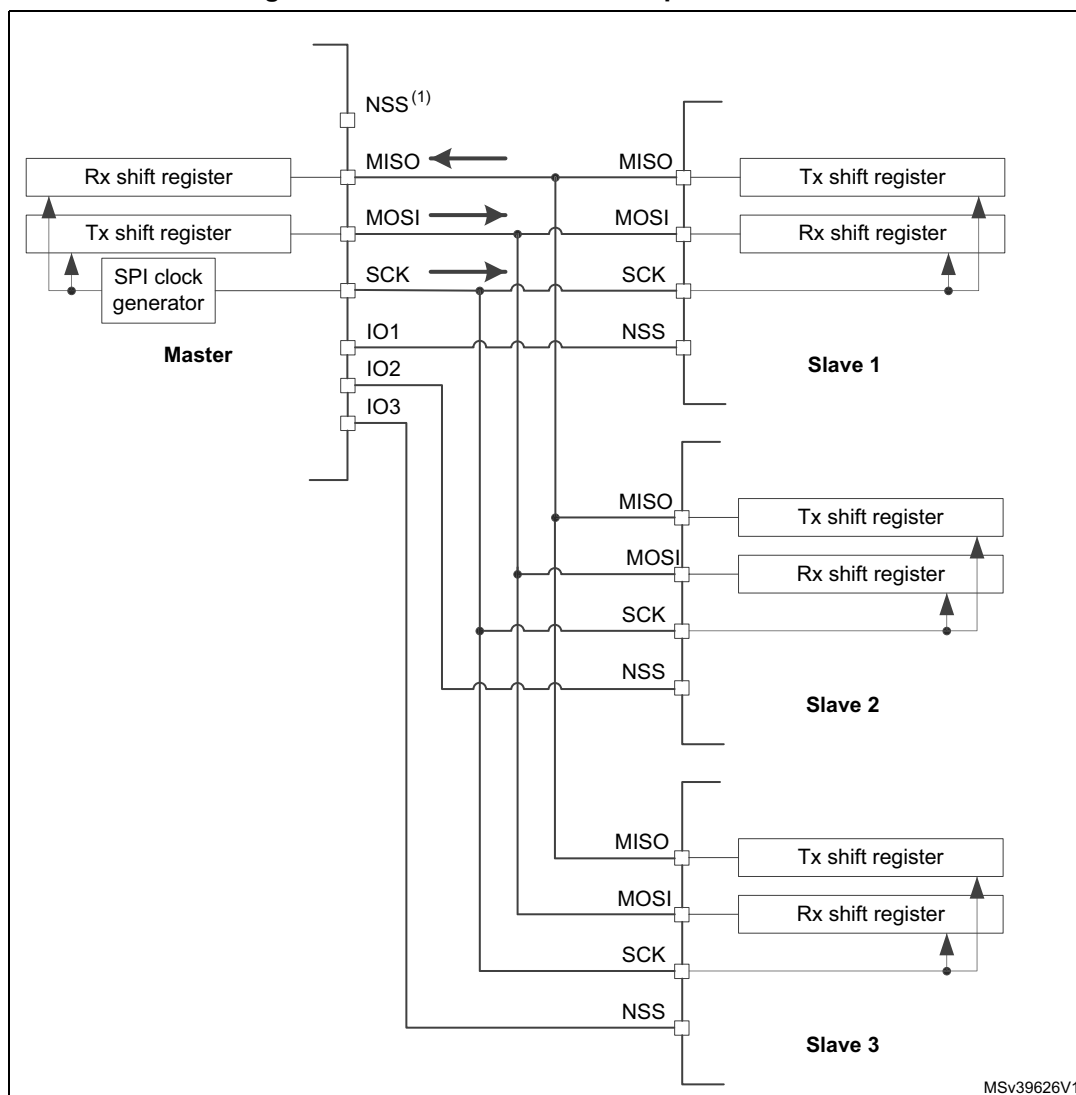
1. The NSS pins can be used to provide a hardware control flow between master and slave. Optionally, the pins can be left unused by the peripheral. Then the flow has to be handled internally for both master and slave. For more details see [Section 35.5.5: Slave select \(NSS\) pin management](#).
2. An 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).*

### 35.5.3 Standard multislave communication

In a configuration with two or more independent slaves, the master uses GPIO pins to manage the chip select lines for each slave (see [Figure 359](#)). 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 359. 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)).

### 35.5.4 Multimaster communication

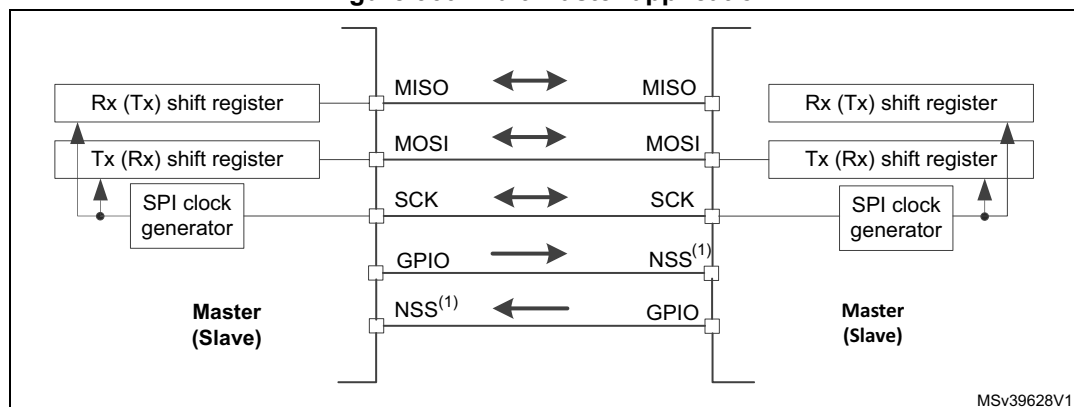
Unless SPI bus is not designed for a multimaster capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, NSS pin is used configured at hardware input mode.

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 360. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.

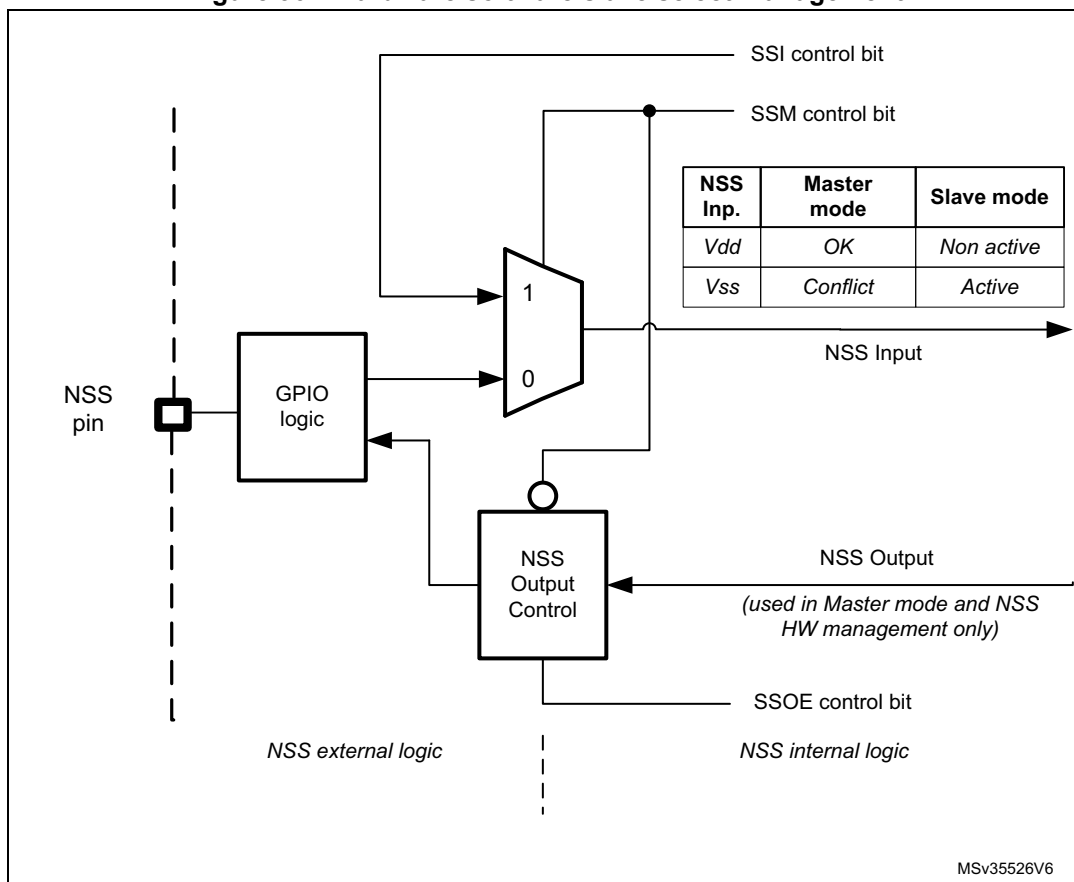
### 35.5.5 Slave select (NSS) pin management

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 361. Hardware/software slave select management



### 35.5.6 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.

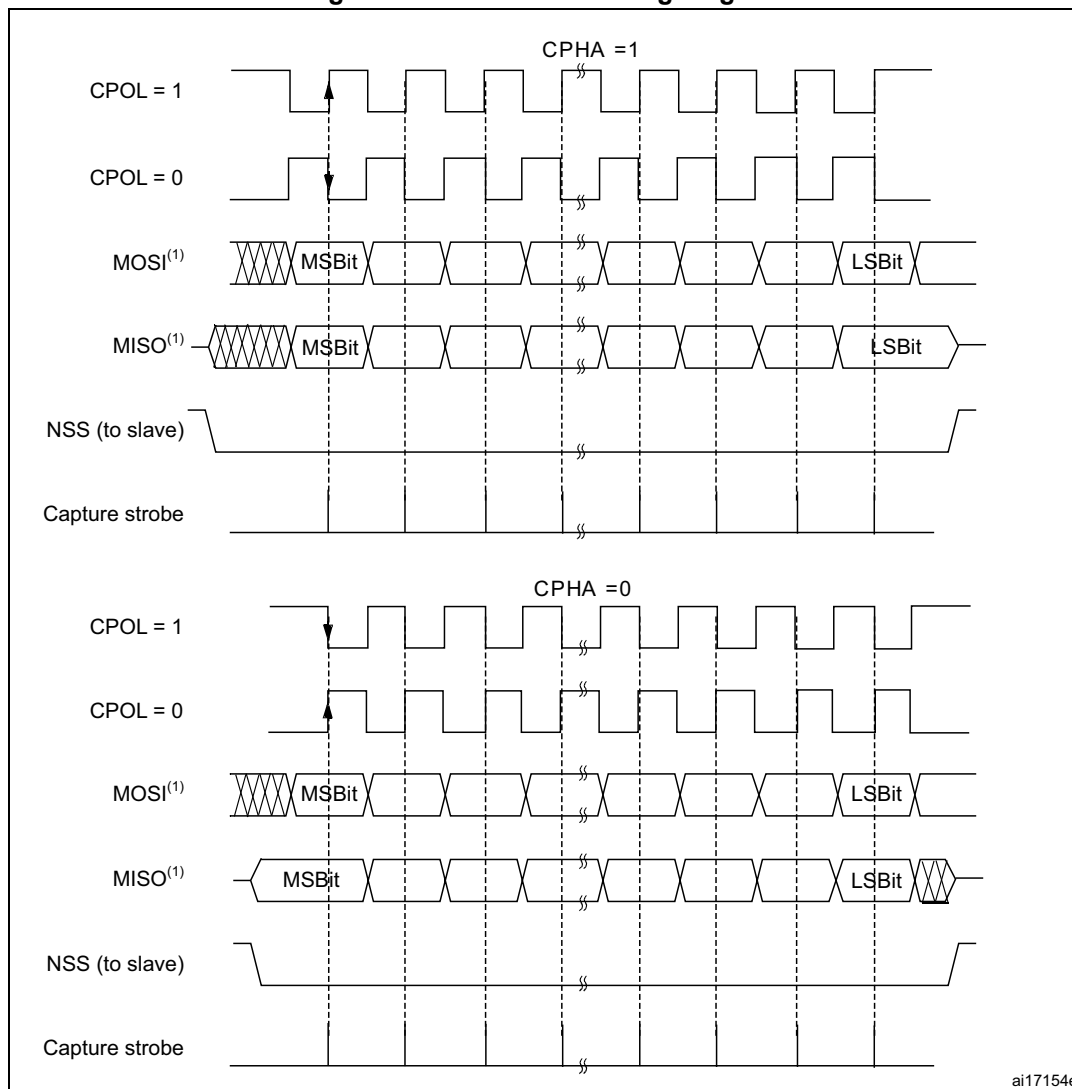
The combination of CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.



Figure 362, 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).

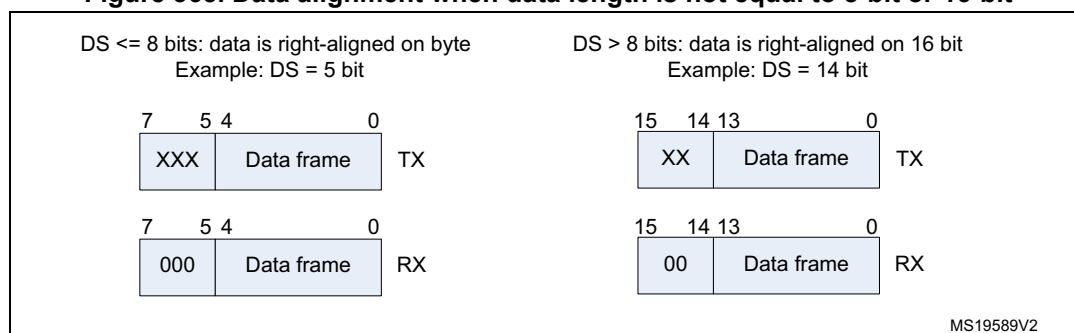
Figure 362. Data clock timing diagram



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 363). During communication, only bits within the data frame are clocked and transferred.

**Figure 363. 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.

### 35.5.7 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 FRXTH bit. The RXFIFO threshold must be aligned to the read access size for the SPIx\_DR register.
  - f) Initialize LDMA\_TX and LDMA\_RX bits if DMA is used in packed mode.
4. Write to SPI\_CRCPR register: Configure the CRC polynomial if needed.
5. Write proper DMA registers: Configure DMA streams dedicated for SPI Tx and Rx in DMA registers if the DMA streams are used.

- Note:
- (1) Step is not required in slave mode.
  - (2) Step is not required in TI mode.
  - (3) Step is not required in NSSP mode.
  - (4) The step is not required in slave mode except slave working at TI mode

### 35.5.8 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.

### 35.5.9 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 35.5.14: 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 35.5.13: 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 365](#) through [Figure 368](#).

Another way to manage the data exchange is to use DMA (see [Communication using DMA \(direct memory addressing\)](#)).

If the next data is received when the RXFIFO is full, an overrun event occurs (see description of OVR flag at [Section 35.5.10: SPI status flags](#)). An overrun event can be polled or handled by an interrupt.

The BSY bit being set indicates ongoing transaction of a current data frame. When the clock signal runs continuously, the BSY flag stays set between data frames at master but becomes low for a minimum duration of one SPI clock at slave between each data frame transfer.

### Sequence handling

A few data frames can be passed at single sequence to complete a message. When transmission is enabled, a sequence begins and continues while any data is present in the TXFIFO of the master. The clock signal is provided continuously by the master until TXFIFO becomes empty, then it stops waiting for additional data.

In receive-only modes, half-duplex (BIDIMODE=1, BIDIOE=0) or simplex (BIDIMODE=0, RXONLY=1) the master starts the sequence immediately when both SPI is enabled and receive-only mode is activated. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled by the master. The master receives data frames continuously up to this moment.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays. Be aware there is no underflow error signal for master or slave in SPI mode, and data from the slave is always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are shorter and bus rate is high.

Each sequence must be encased by the NSS pulse in parallel with the multislave system to select just one of the slaves for communication. In a single slave system it is not necessary to control the slave with NSS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. NSS can be managed by both software and hardware (see [Section 35.5.5: 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.*

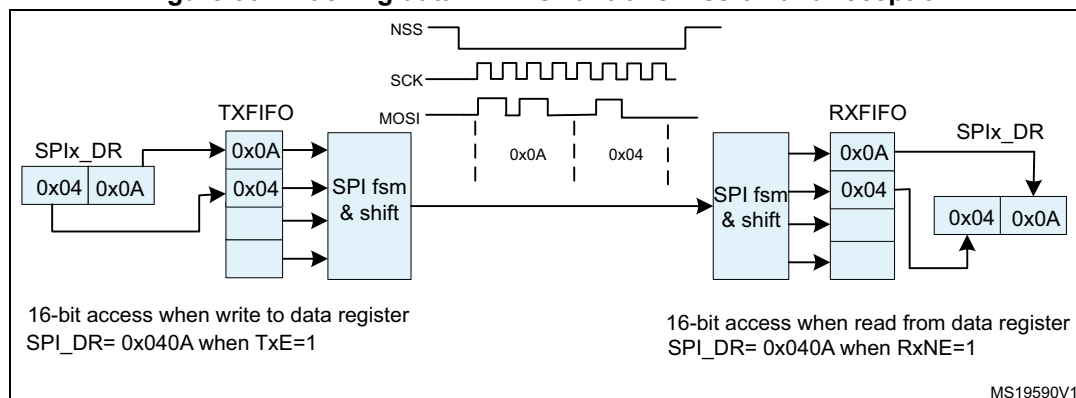
### Data packing

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 364](#) 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 364. 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 365](#) through [Figure 368](#).

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 1137.](#))



## Communication diagrams

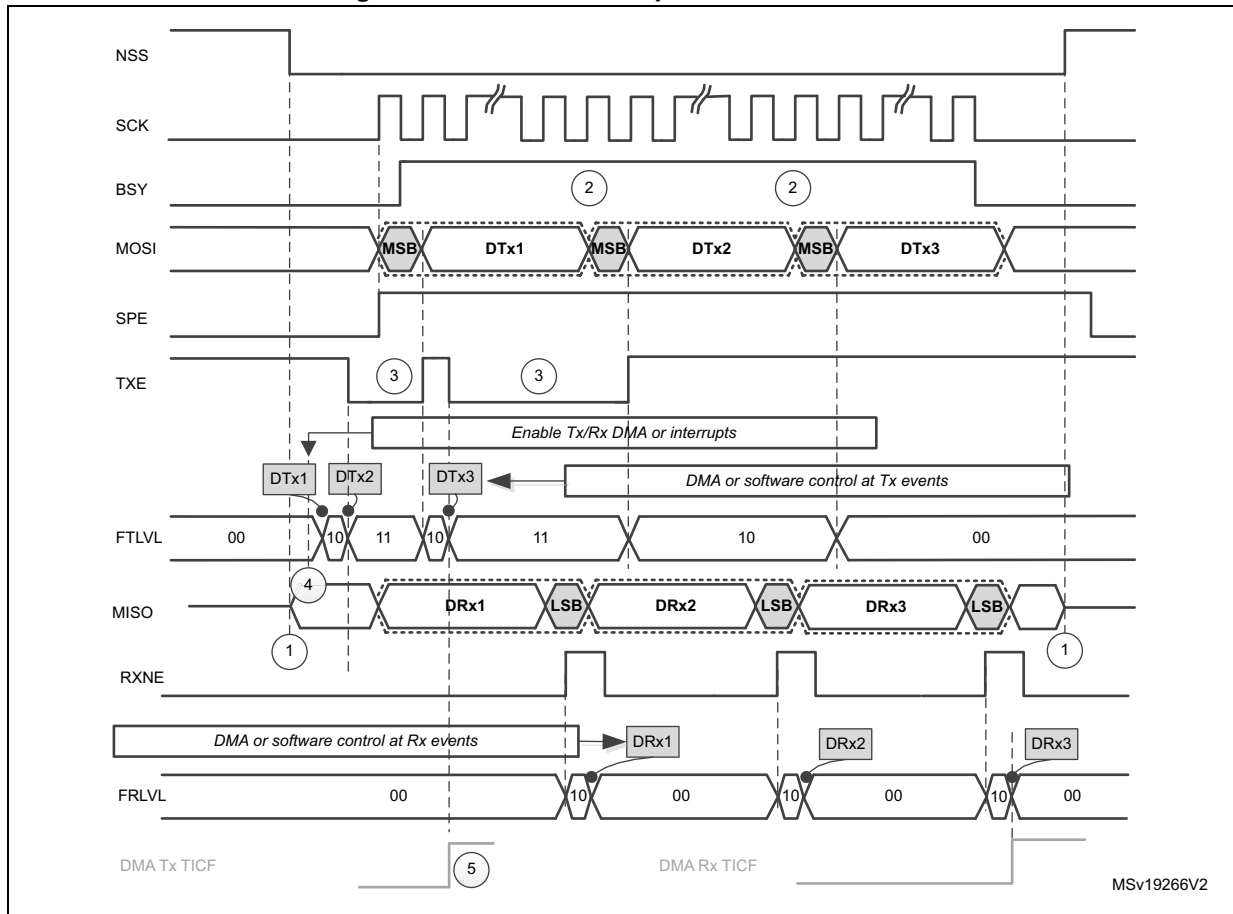
Some typical timing schemes are explained in this section. These schemes are valid no matter if the SPI events are handled by polling, interrupts or DMA. For simplicity, the LSBFIRST=0, CPOL=0 and CPHA=1 setting is used as a common assumption here. No complete configuration of DMA streams is provided.

The following numbered notes are common for [Figure 365 on page 1141](#) through [Figure 368 on page 1144](#):

1. The slave starts to control MISO line as NSS is active and SPI is enabled, and is disconnected from the line when one of them is released. Sufficient time must be provided for the slave to prepare data dedicated to the master in advance before its transaction starts.  
At the master, the SPI peripheral takes control at MOSI and SCK signals (occasionally at NSS signal as well) only if SPI is enabled. If SPI is disabled the SPI peripheral is disconnected from GPIO logic, so the levels at these lines depends on GPIO setting exclusively.
2. At the master, BSY stays active between frames if the communication (clock signal) is continuous. At the slave, BSY signal always goes down for at least one clock cycle between data frames.
3. The TXE signal is cleared only if TXFIFO is full.
4. The DMA arbitration process starts just after the TXDMAEN bit is set. The TXE interrupt is generated just after the TXEIE is set. As the TXE signal is at an active level, data transfers to TxFIFO start, until TxFIFO becomes full or the DMA transfer completes.
5. If all the data to be sent can fit into TxFIFO, the DMA Tx TCIF flag can be raised even before communication on the SPI bus starts. This flag always rises before the SPI transaction is completed.
6. The CRC value for a package is calculated continuously frame by frame in the SPIx\_TXCRCR and SPIx\_RXCRCR registers. The CRC information is processed after the entire data package has completed, either automatically by DMA (Tx channel must be set to the number of data frames to be processed) or by SW (the user must handle CRCNEXT bit during the last data frame processing).  
While the CRC value calculated in SPIx\_TXCRCR is simply sent out by transmitter, received CRC information is loaded into RxFIFO and then compared with the SPIx\_RXCRCR register content (CRC error flag can be raised here if any difference). This is why the user must take care to flush this information from the FIFO, either by software reading out all the stored content of RxFIFO, or by DMA when the proper number of data frames is preset for Rx channel (number of data frames + number of CRC frames) (see the settings at the example assumption).
7. In data packed mode, TxE and RxNE events are paired and each read/write access to the FIFO is 16 bits wide until the number of data frames are even. If the TxFIFO is  $\frac{3}{4}$  full FTLVL status stays at FIFO full level. That is why the last odd data frame cannot be stored before the TxFIFO becomes  $\frac{1}{2}$  full. This frame is stored into TxFIFO with an 8-bit access either by software or automatically by DMA when LDMA\_TX control is set.
8. To receive the last odd data frame in packed mode, the Rx threshold must be changed to 8-bit when the last data frame is processed, either by software setting FRXTH=1 or automatically by a DMA internal signal when LDMA\_RX is set.



Figure 365. Master full-duplex communication



Assumptions for master full-duplex communication example:

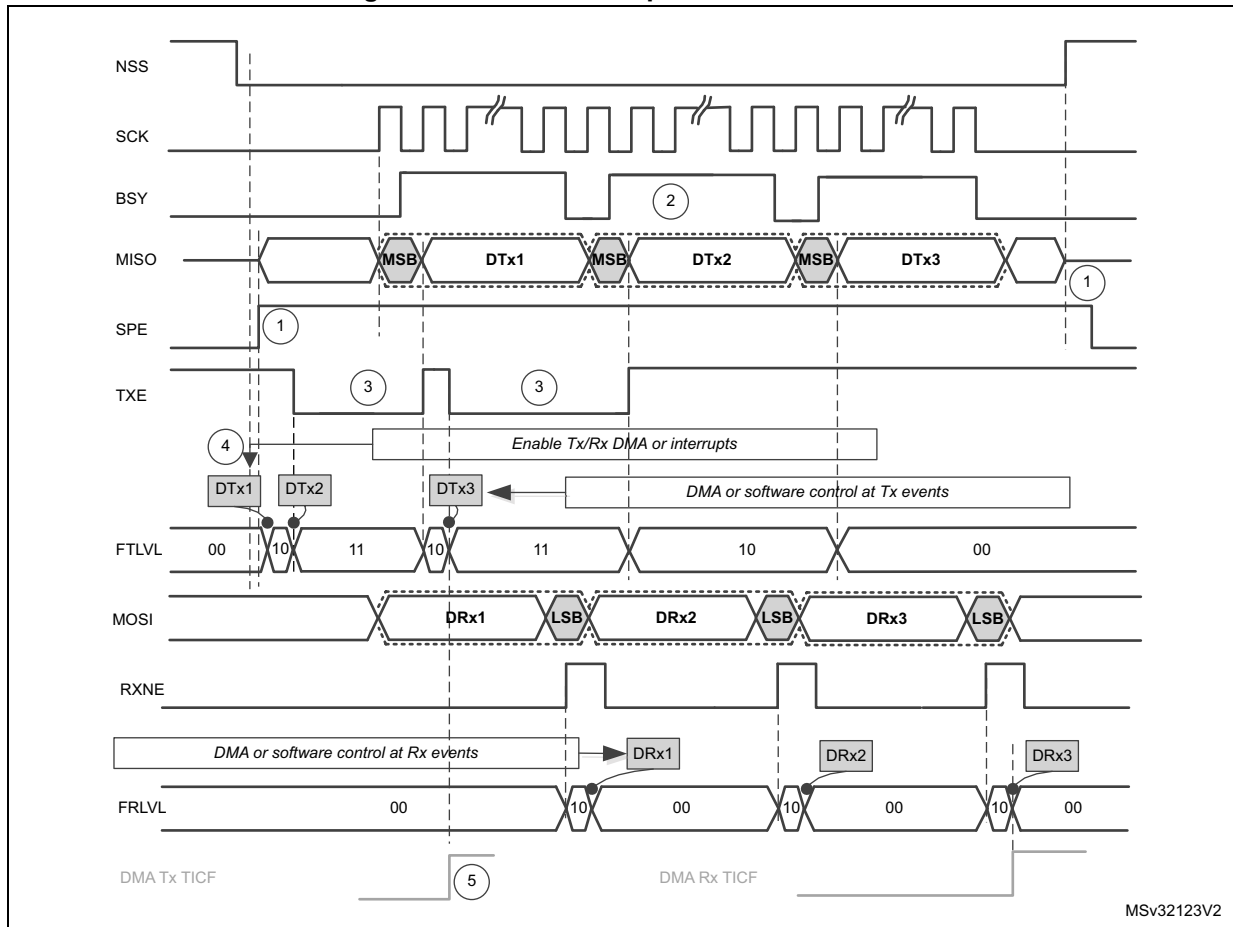
- Data size > 8 bit

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 1140](#) for details about common assumptions and notes.

Figure 366. Slave full-duplex communication



Assumptions for slave full-duplex communication example:

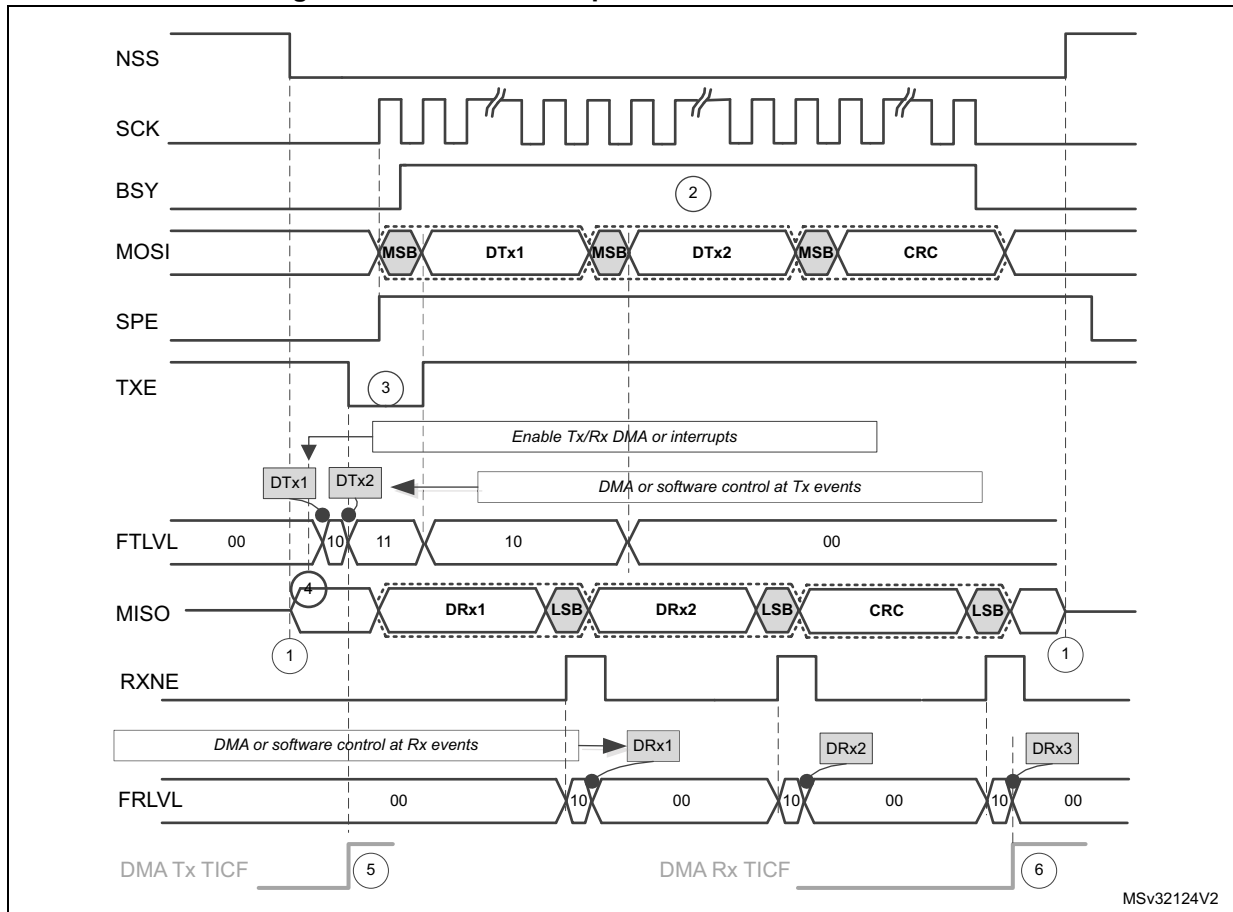
- Data size > 8 bit

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 1140](#) for details about common assumptions and notes.

Figure 367. Master full-duplex communication with CRC



Assumptions for master full-duplex communication with CRC example:

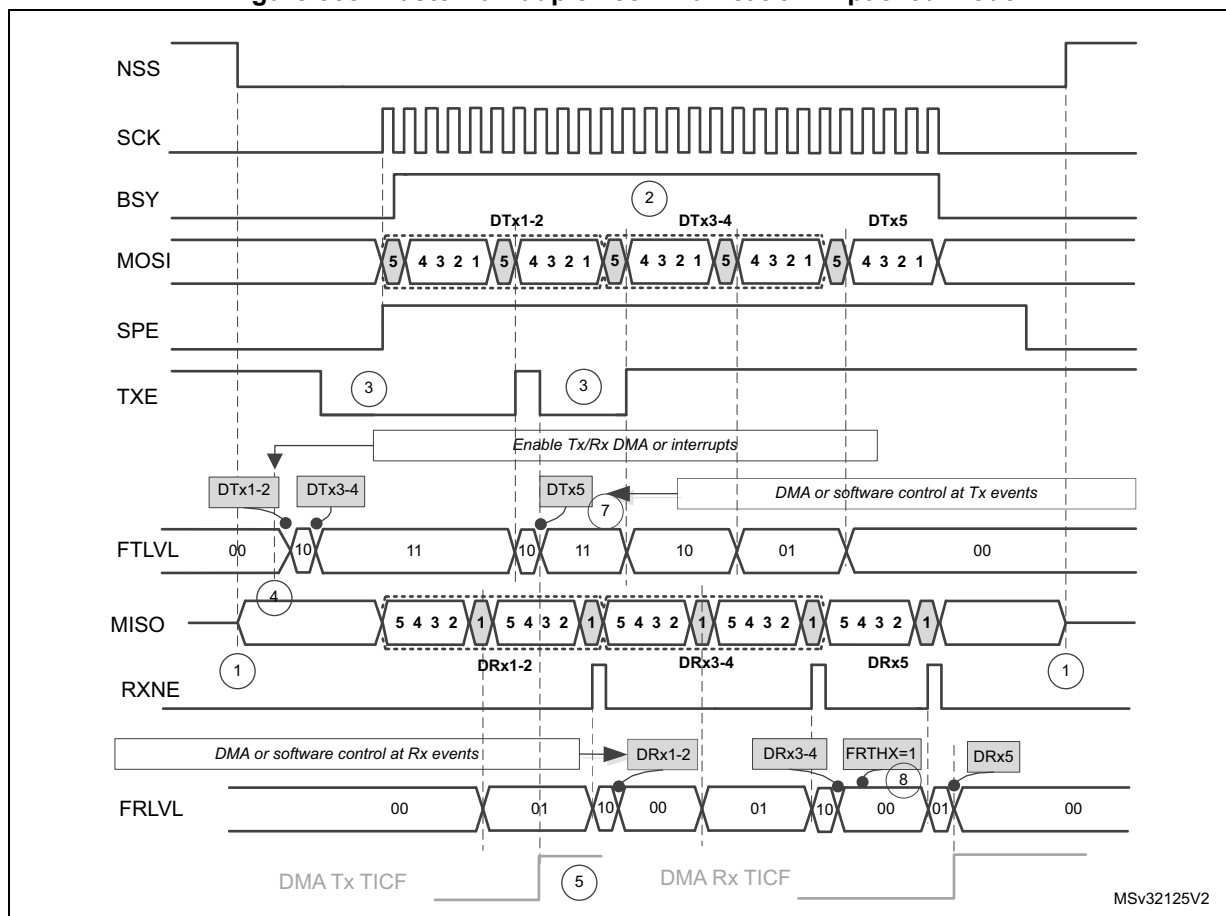
- Data size = 16 bit
- CRC enabled

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 1140](#) for details about common assumptions and notes.

Figure 368. 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 1140](#) for details about common assumptions and notes.

### 35.5.10 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.

### 35.5.11 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 35.5.14: 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.

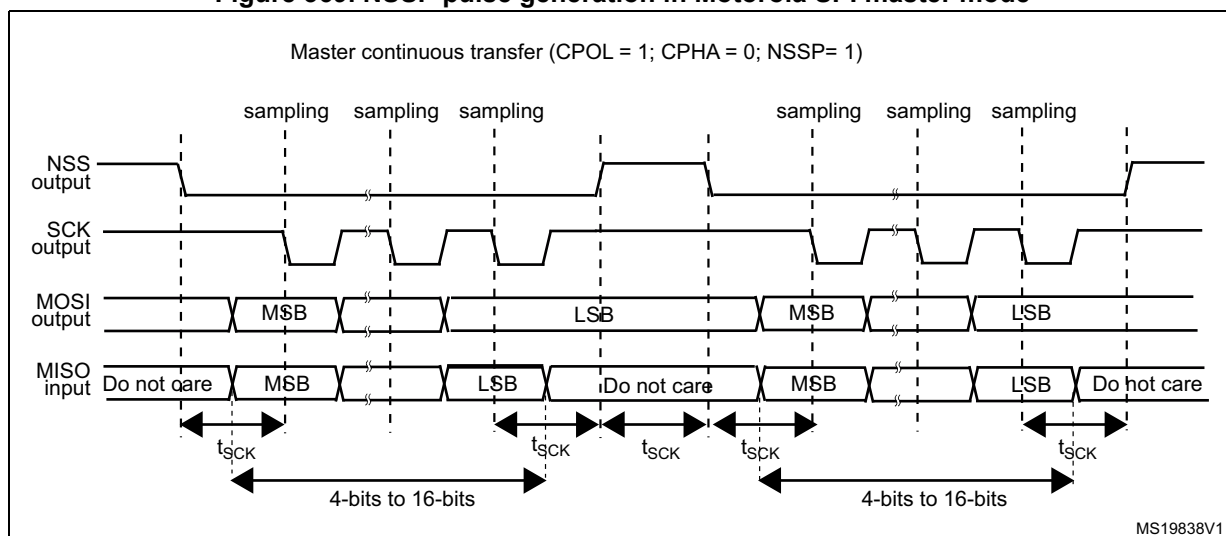
The FRE flag is cleared when SPIx\_SR register is read. If the ERRIE bit is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no longer guaranteed and communications should be reinitiated by the master when the slave SPI is enabled again.

### 35.5.12 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 369 illustrates NSS pin management when NSSP pulse mode is enabled.

Figure 369. NSSP pulse generation in Motorola SPI master mode



Note: Similar behavior is encountered when CPOL = 0. In this case the sampling edge is the rising edge of SCK, and NSS assertion and deassertion refer to this sampling edge.

### 35.5.13 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 370). 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:

$$\frac{t_{\text{baud\_rate}}}{2} + 4 \times t_{\text{pclk}} < t_{\text{release}} < \frac{t_{\text{baud\_rate}}}{2} + 6 \times t_{\text{pclk}}$$

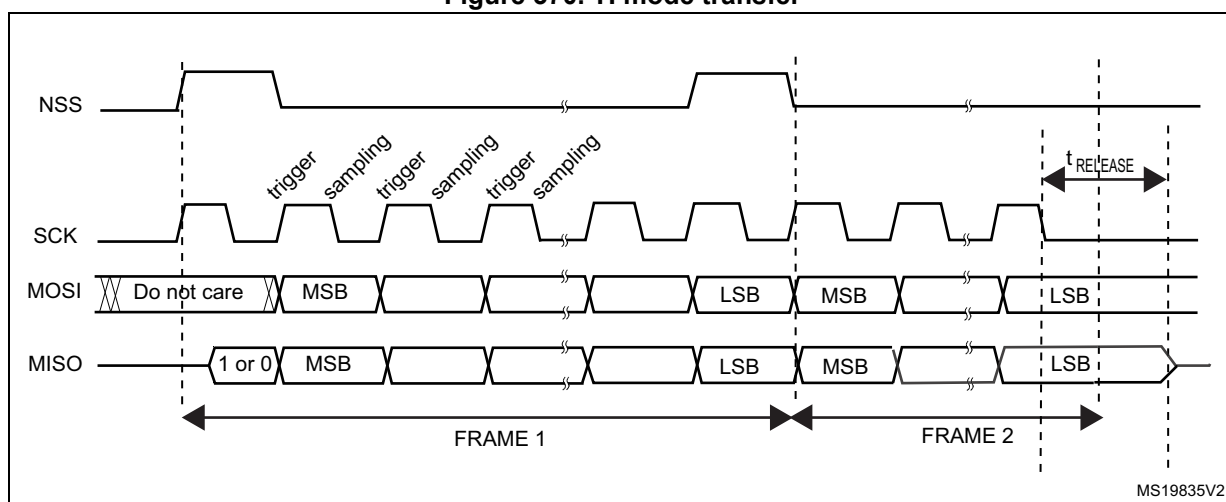
If the slave detects a misplaced NSS pulse during a data frame transaction the TIFRE flag is set.

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 370: TI mode transfer* shows the SPI communication waveforms when TI mode is selected.

**Figure 370. TI mode transfer**



### 35.5.14 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.*

### 35.6 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 206. SPI interrupt requests**

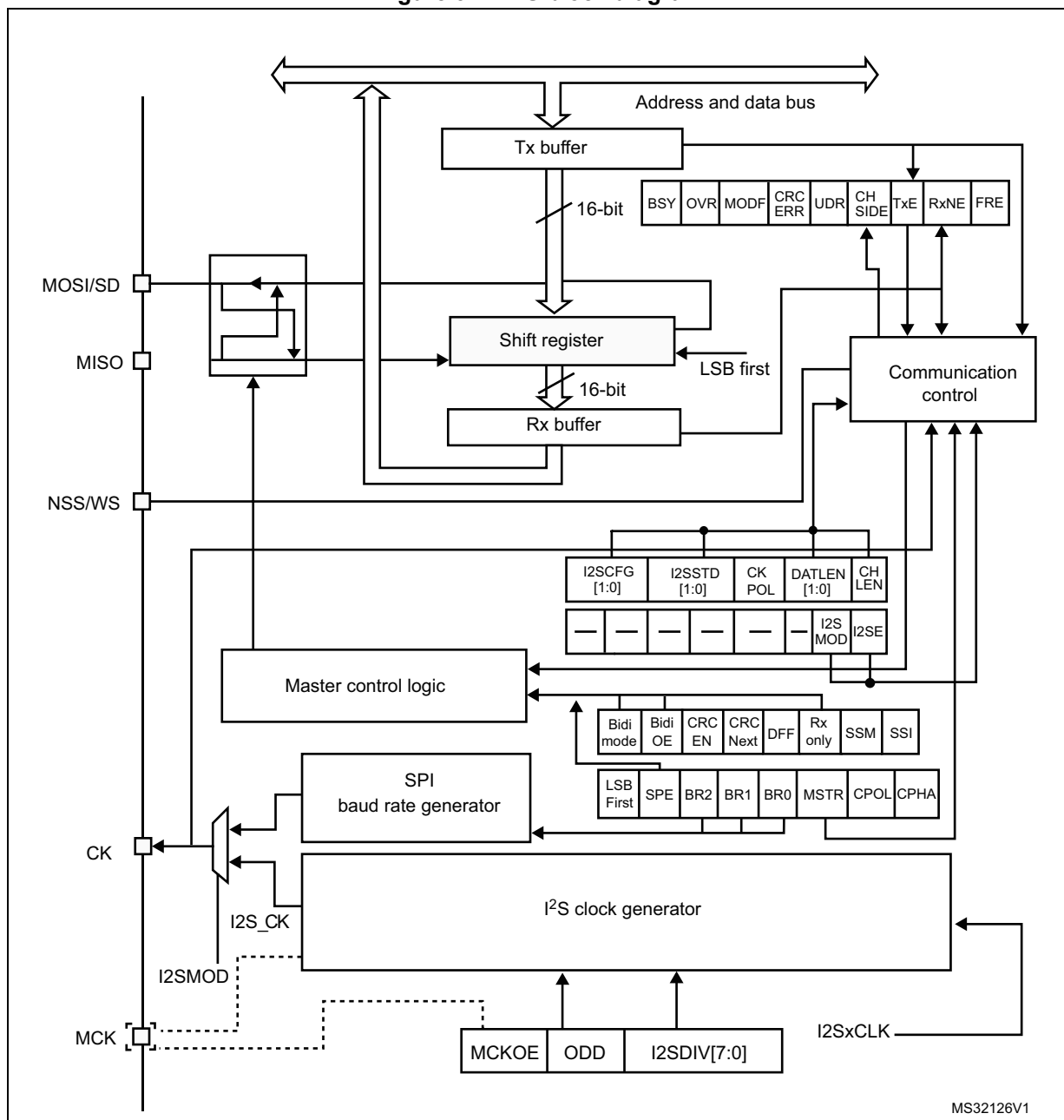
Interrupt event	Event flag	Enable Control bit
Transmit TXFIFO ready to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
TI frame format error	FRE	
CRC protocol error	CRCERR	

### 35.7 I2S functional description

#### 35.7.1 I2S general description

The block diagram of the I2S is shown in [Figure 371](#).

Figure 371. 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.

### 35.7.2 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).

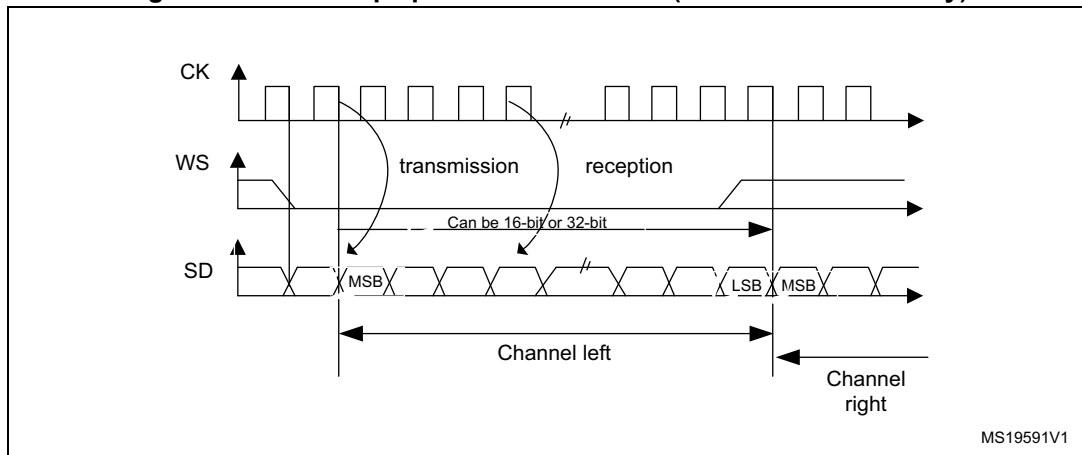
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

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**

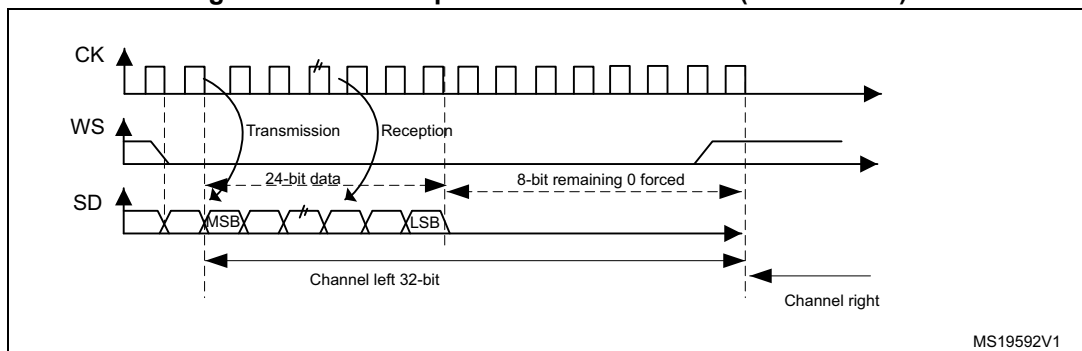
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

**Figure 372. 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.

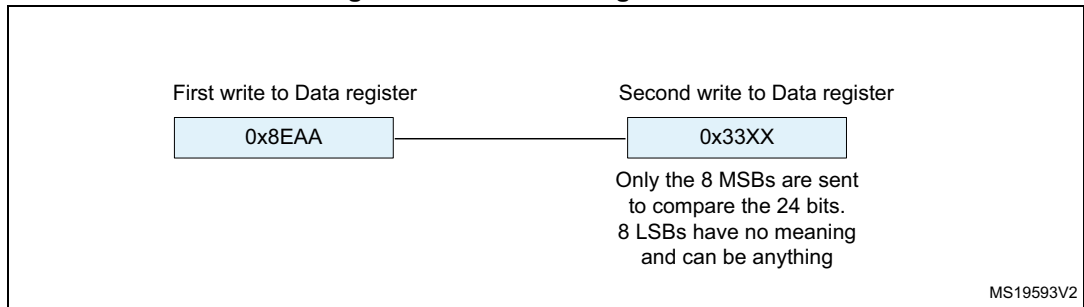
**Figure 373. I<sup>2</sup>S Philips standard waveforms (24-bit frame)**



This mode needs two write or read operations to/from the SPIx\_DR register.

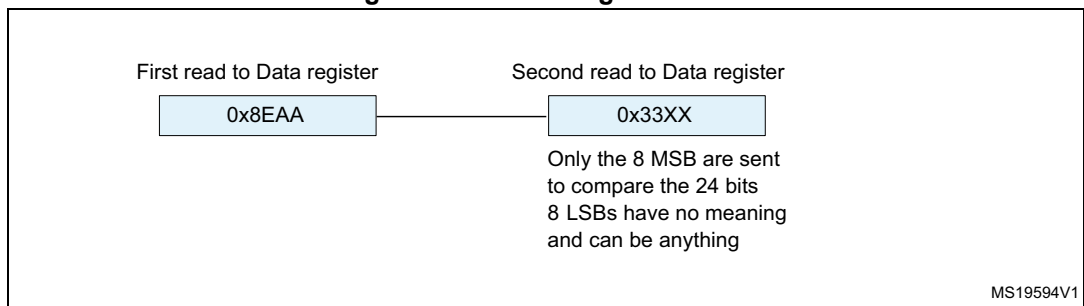
- In transmission mode:  
If 0x8EAA33 has to be sent (24-bit):

**Figure 374. Transmitting 0x8EAA33**

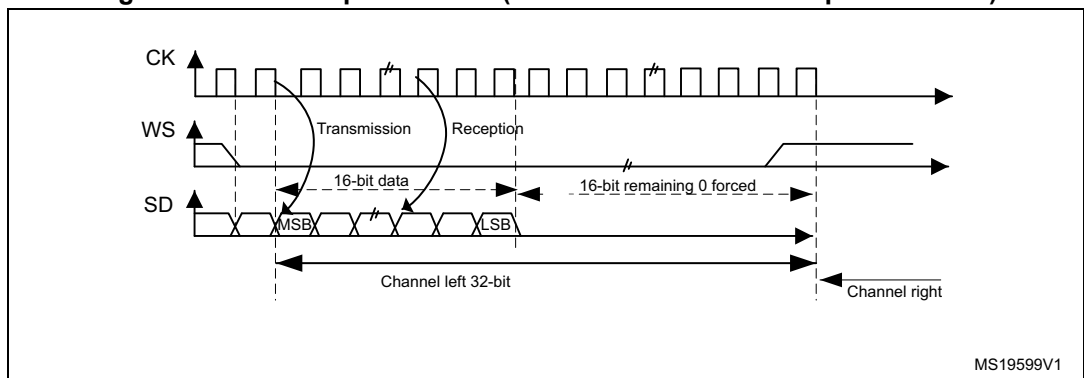


- In reception mode:  
If data 0x8EAA33 is received:

**Figure 375. Receiving 0x8EAA33**



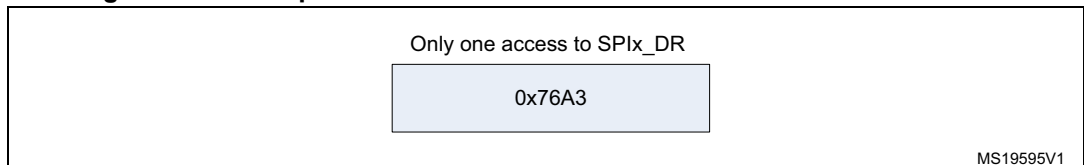
**Figure 376. I<sup>2</sup>S Philips standard (16-bit extended to 32-bit packet frame)**



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 377](#) is required.

**Figure 377. 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.

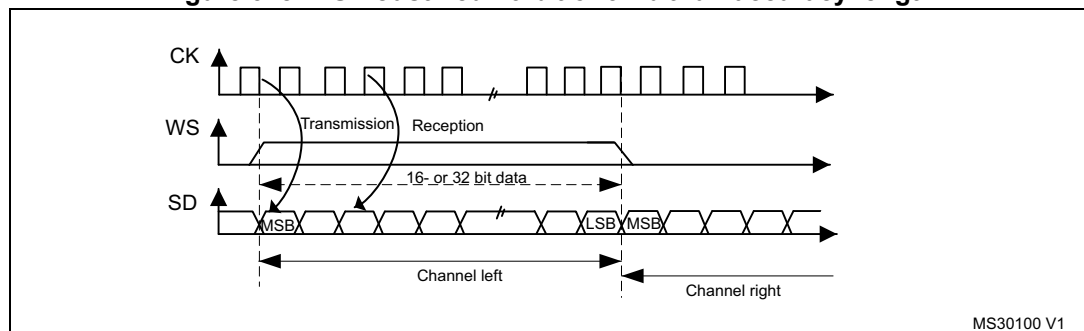
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

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**

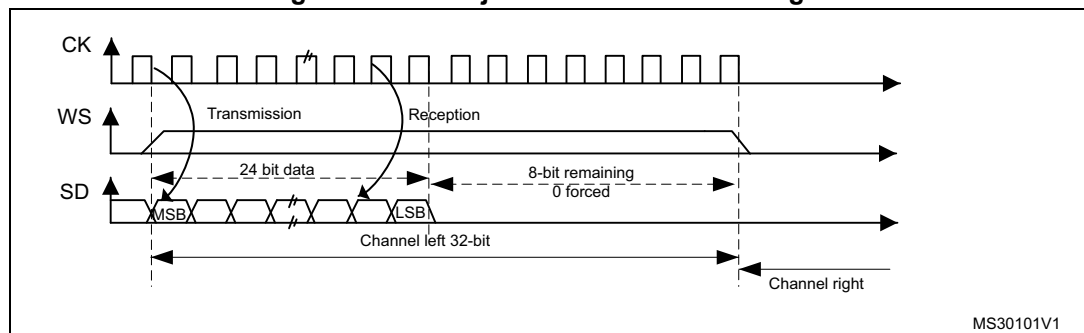
For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

**Figure 378. MSB Justified 16-bit or 32-bit full-accuracy length**

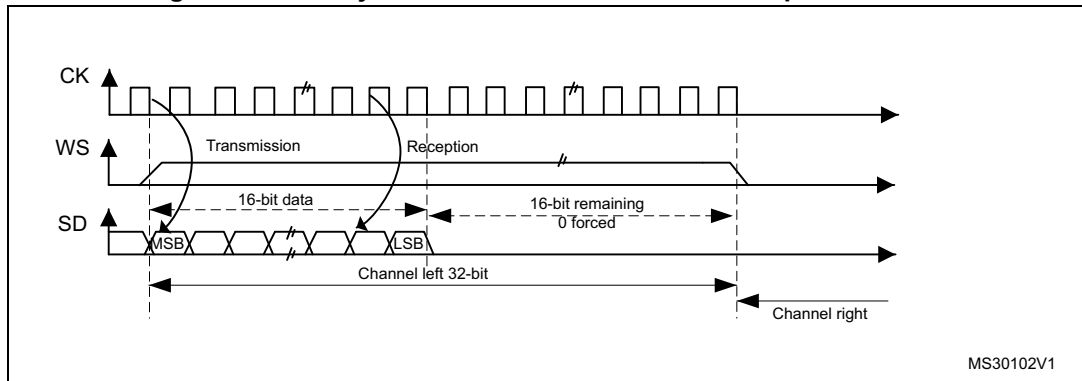


Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

**Figure 379. MSB justified 24-bit frame length**



**Figure 380. MSB justified 16-bit extended to 32-bit packet frame**

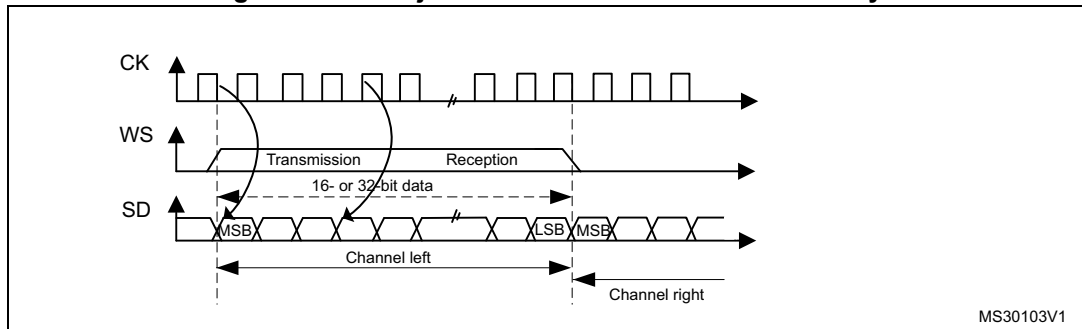


**LSB justified standard**

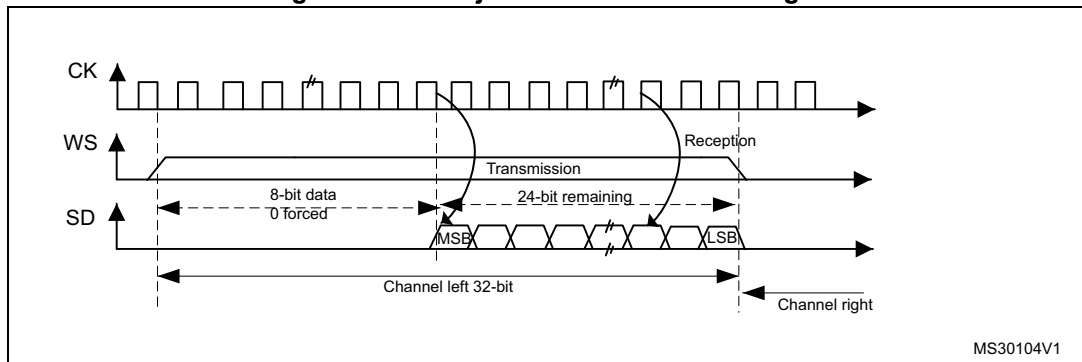
This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

The sampling of the input and output signals is the same as for the I<sup>2</sup>S Philips standard.

**Figure 381. LSB justified 16-bit or 32-bit full-accuracy**



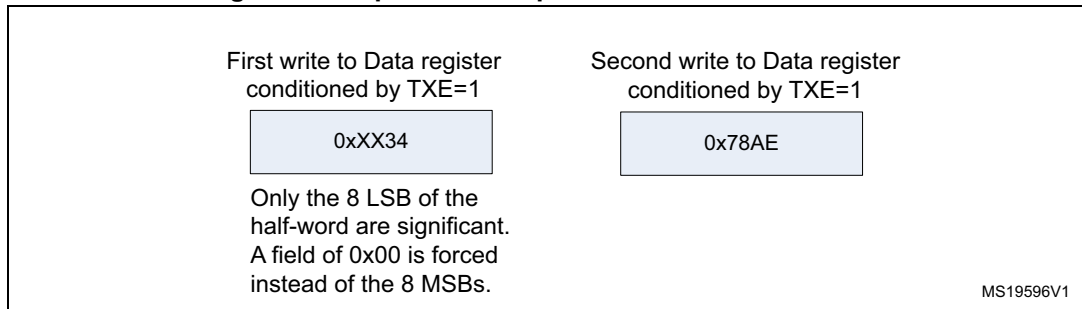
**Figure 382. LSB justified 24-bit frame length**



- 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.

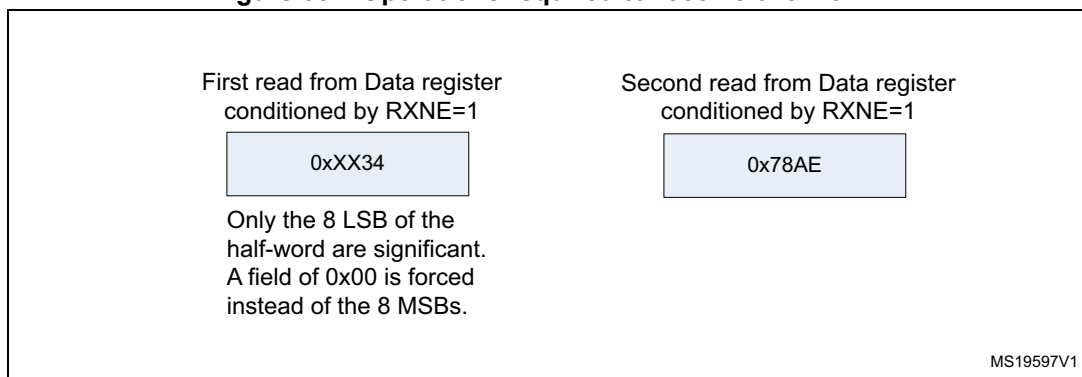


**Figure 383. Operations required to transmit 0x3478AE**

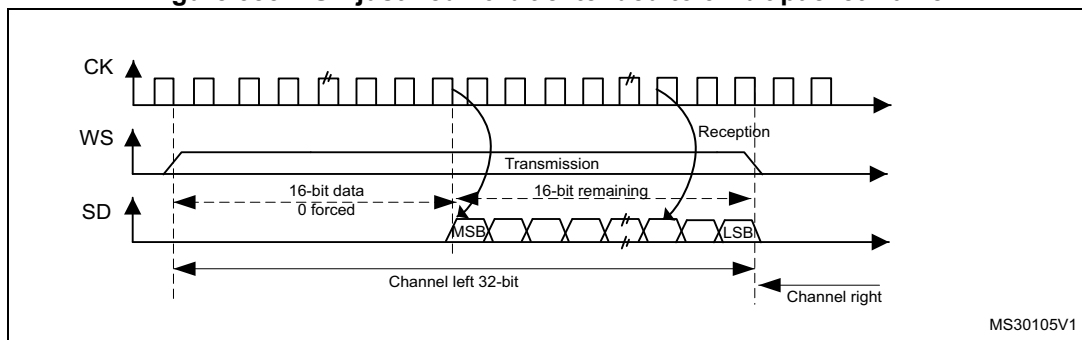


- In reception mode:  
If data 0x3478AE are received, two successive read operations from the SPIx\_DR register are required on each RXNE event.

**Figure 384. Operations required to receive 0x3478AE**



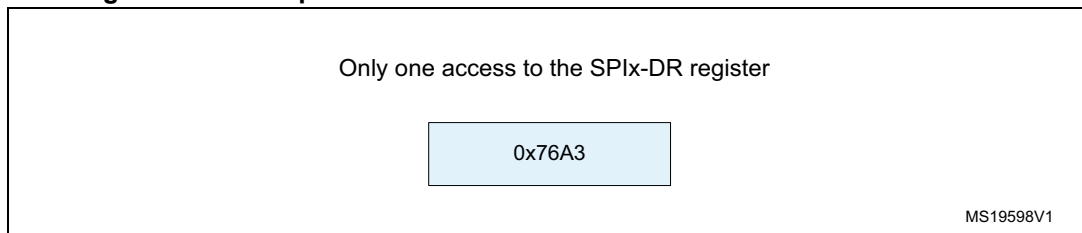
**Figure 385. LSB justified 16-bit extended to 32-bit packet frame**



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 386](#) is required.

**Figure 386. 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.

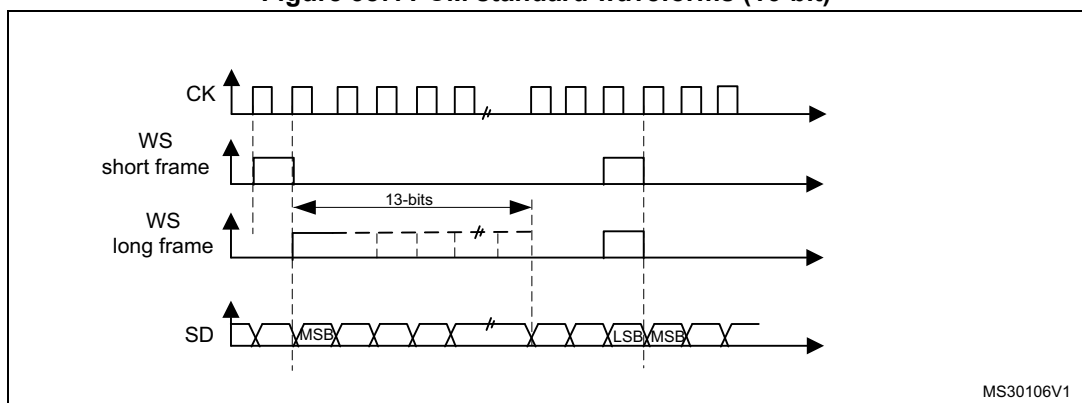
**PCM standard**

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.

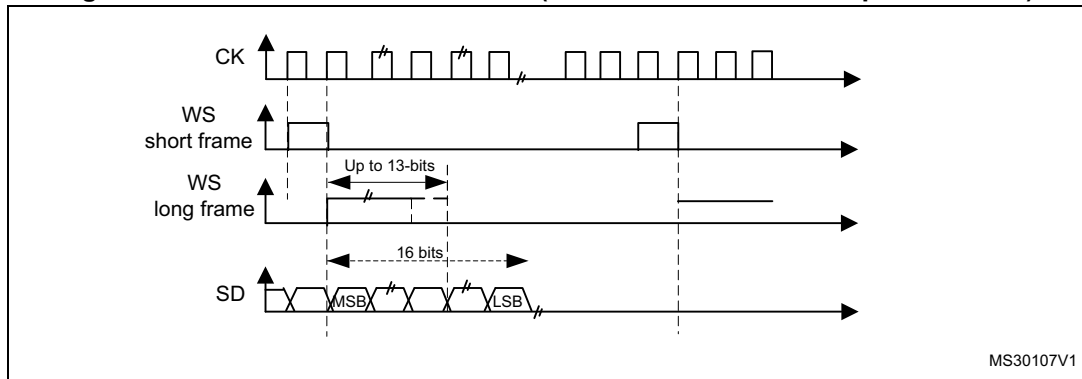
**Figure 387. PCM standard waveforms (16-bit)**



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 388. PCM standard waveforms (16-bit extended to 32-bit packet frame)**



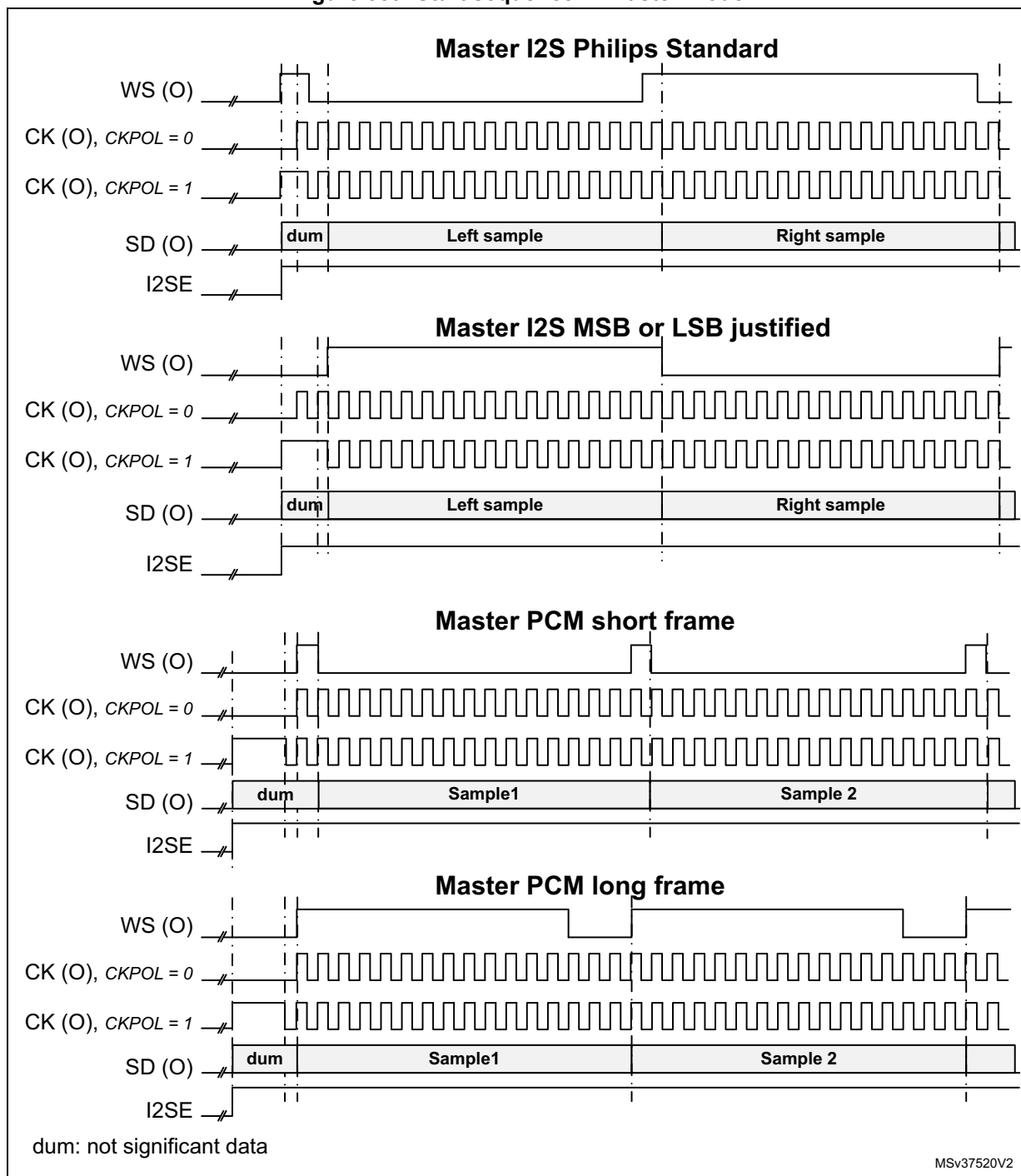
MS30107V1

*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.

### 35.7.3 Start-up description

The [Figure 389](#) 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 389. 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.

### 35.7.4 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.

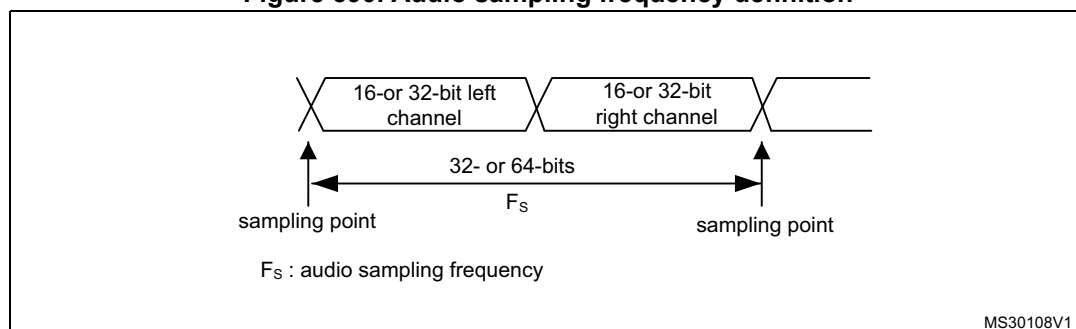
I<sup>2</sup>S bit rate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I<sup>2</sup>S bit rate is calculated as follows:

$$I^2S \text{ bit rate} = 16 \times 2 \times f_s$$

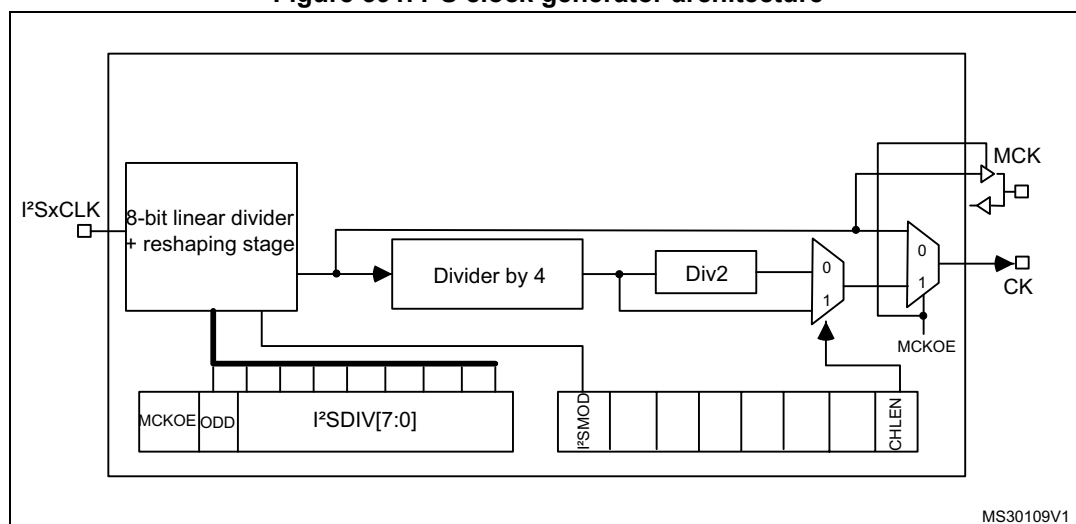
It is: I<sup>2</sup>S bit rate = 32 × 2 × f<sub>s</sub> if the packet length is 32-bit wide.

Figure 390. Audio sampling frequency definition



When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 391. I<sup>2</sup>S clock generator architecture



1. Where x can be 2 or 3.

*Figure 391* 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: I2SDIV 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 207. Audio-frequency precision using 48 MHz clock derived from HSE<sup>(1)</sup>**

SYSCLK (MHz)	Data length	I2SDIV	I2SODD	MCLK	Target fs (Hz)	Real fs (kHz)	Error
48	16	8	0	No	96000	93750	2.3438%
48	32	4	0	No	96000	93750	2.3438%
48	16	15	1	No	48000	48387.0968	0.8065%
48	32	8	0	No	48000	46875	2.3438%
48	16	17	0	No	44100	44117.647	0.0400%
48	32	8	1	No	44100	44117.647	0.0400%
48	16	23	1	No	32000	31914.8936	0.2660%
48	32	11	1	No	32000	32608.696	1.9022%
48	16	34	0	No	22050	22058.8235	0.0400%
48	32	17	0	No	22050	22058.8235	0.0400%
48	16	47	0	No	16000	15957.4468	0.2660%
48	32	23	1	No	16000	15957.447	0.2660%
48	16	68	0	No	11025	11029.4118	0.0400%
48	32	34	0	No	11025	11029.412	0.0400%
48	16	94	0	No	8000	7978.7234	0.2660%
48	32	47	0	No	8000	7978.7234	0.2660%
48	16	2	0	Yes	48000	46875	2.3430%
48	32	2	0	Yes	48000	46875	2.3430%
48	16	2	0	Yes	44100	46875	6.2925%
48	32	2	0	Yes	44100	46875	6.2925%
48	16	3	0	Yes	32000	31250	2.3438%
48	32	3	0	Yes	32000	31250	2.3438%
48	16	4	1	Yes	22050	20833.333	5.5178%
48	32	4	1	Yes	22050	20833.333	5.5178%
48	16	6	0	Yes	16000	15625	2.3438%
48	32	6	0	Yes	16000	15625	2.3438%
48	16	8	1	Yes	11025	11029.4118	0.0400%
48	32	8	1	Yes	11025	11029.4118	0.0400%
48	16	11	1	Yes	8000	8152.17391	1.9022%
48	32	11	1	Yes	8000	8152.17391	1.9022%

1. This table gives only example values for different clock configurations. Other configurations allowing optimum clock precision are possible.

### 35.7.5 I<sup>2</sup>S master mode

The I2S can be configured in master mode. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, controlled by the MCKOE bit in the SPIx\_I2SPR register.

#### Procedure

1. Select the I2SDIV[7:0] bits in the SPIx\_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPIx\_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPIx\_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to [Section 35.7.4: Clock generator](#)).
3. Set the I2SMOD bit in the SPIx\_I2SCFGR register to activate the I2S functions and choose the I<sup>2</sup>S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I<sup>2</sup>S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPIx\_I2SCFGR register.
4. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPIx\_CR2 register.
5. The I2SE bit in SPIx\_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPIx\_I2SPR is set.

#### Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Lets assume the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the right channel have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPIx\_CR2 register is set.

For more details about the write operations depending on the I<sup>2</sup>S standard mode selected, refer to [Section 35.7.2: 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 I2S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

### Reception sequence

The operating mode is the same as for transmission mode except for the point 3 (refer to the procedure described in [Section 35.7.5: 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 I2S cell.

For more details about the read operations depending on the I<sup>2</sup>S standard mode selected, refer to [Section 35.7.2: Supported audio protocols](#).

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPIx\_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S, specific actions are required to ensure that the I2S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  - a) Wait for the second to last RXNE = 1 (n – 1)
  - b) Then wait 17 I2S clock cycles (using a software loop)
  - c) Disable the I2S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, 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 I2S clock cycle (using a software loop)
  - c) Disable the I2S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I2S:
  - a) Wait for the second to last RXNE = 1 (n – 1)
  - b) Then wait one I2S clock cycle (using a software loop)
  - c) Disable the I2S (I2SE = 0)

*Note:* The BSY flag is kept low during transfers.

### 35.7.6 I<sup>2</sup>S slave mode

For the slave configuration, the I2S can be configured in transmission or reception mode. The operating mode is following mainly the same rules as described for the 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 35.7.2: 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 35.7.6: 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 35.7.2: Supported audio protocols](#).

If data are received while the preceding received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPIx\_CR2 register, an interrupt is generated to indicate the error.

To switch off the I2S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

*Note:* The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

### 35.7.7 I2S status flags

Three status flags are provided for the application to fully monitor the state of the I2S bus.

#### Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I2S.

When BSY is set, it indicates that the I2S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software needs to disable the I2S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I2S is in master receiver mode.

The BSY flag is cleared:

- When a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- When the I2S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I2S clock cycle between each transfer

*Note:* Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

**Tx buffer empty flag (TXE)**

When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I2S is disabled (I2SE bit is reset).

**RX buffer not empty (RXNE)**

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPIx\_DR register is read.

**Channel Side flag (CHSIDE)**

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I2S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPIx\_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I2S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

When the OVR or UDR flag in the SPIx\_SR is set and the ERRIE bit in SPIx\_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPIx\_SR status register (once the interrupt source has been cleared).

**35.7.8 I2S error flags**

There are three error flags for the I2S cell.

**Underrun flag (UDR)**

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPIx\_DR. It is available when the I2SMOD bit in the SPIx\_I2SCFGR register is set. An interrupt may be generated if the ERRIE bit in the SPIx\_CR2 register is set.

The UDR bit is cleared by a read operation on the SPIx\_SR register.

**Overrun flag (OVR)**

This flag is set when data are received and the previous data have not yet been read from the SPIx\_DR register. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in the SPIx\_CR2 register.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPIx\_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPIx\_DR register followed by a read access to the SPIx\_SR register.

**Frame error flag (FRE)**

This flag can be set by hardware only if the I2S is configured in Slave mode. It is set if the external master is changing the WS line while the slave is not expecting this change. If the

synchronization is lost, the following steps are required to recover from this state and resynchronize the external master device with the I2S slave device:

1. Disable the I2S.
2. Enable it again when the correct level is detected on the WS line (WS line is high in 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.

### 35.7.9 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.

## 35.8 I2S interrupts

[Table 208](#) provides the list of I2S interrupts.

**Table 208. I2S 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	

## 35.9 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.

### 35.9.1 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 **BIDIMODE**: Bidirectional data mode enable.

This bit enables half-duplex communication using common single bidirectional data line. Keep RXONLY bit clear when bidirectional mode is active.

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

*Note: This bit is not used in I<sup>2</sup>S mode.*

Bit 14 **BIDIOE**: Output enable in bidirectional mode

This bit combined with the BIDIMODE bit selects the direction of transfer in bidirectional mode.

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

*Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.*

*This bit is not used in I<sup>2</sup>S mode.*

Bit 13 **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 1136](#).  
This bit is not used in I<sup>2</sup>S mode.*
- Bits 5:3 **BR[2:0]**: Baud rate control  
000:  $f_{PCLK}/2$   
001:  $f_{PCLK}/4$   
010:  $f_{PCLK}/8$   
011:  $f_{PCLK}/16$   
100:  $f_{PCLK}/32$   
101:  $f_{PCLK}/64$   
110:  $f_{PCLK}/128$   
111:  $f_{PCLK}/256$   
*Note: These bits should not be changed when communication is ongoing.  
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.*

### 35.9.2 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 1136 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 1136 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

### 35.9.3 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 35.5.11: SPI error flags](#) and [Section 35.7.8: 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 35.5.10: SPI status flags and Procedure for disabling the SPI on page 1136](#).*
- 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 1168](#) 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 1146](#) 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 1168](#) 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

### 35.9.4 SPI data register (SPIx\_DR)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:0 **DR[15:0]**: Data register

Data received or to be transmitted

The data register serves as an interface between the Rx and Tx FIFOs. When the data register is read, RxFIFO is accessed while the write to data register accesses Tx FIFO (See [Section 35.5.9: 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.*

### 35.9.5 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.*

### 35.9.6 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.*

### 35.9.7 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.*

### 35.9.8 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.	ASTRTEN	I2SMOD	I2SE	I2SCFG[1:0]	PCMSYNC	Res.	I2SSTD[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 **ASTRTEN**: 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 35.7.3: 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 35.7.2 on page 1152](#)

*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.*

### 35.9.9 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]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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 35.7.3 on page 1159](#).

*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 35.7.3 on page 1159](#).

*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.*



35.9.10 SPI/I2S register map

Table 209 shows the SPI/I2S register map and reset values.

Table 209. 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	BIDIOE	CRCEN	CRCNEXT	CRCL	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPIx_CR2	Res.	LDMA_TX	LDMA_RX	FRXTH	DS[3:0]			TXEIE	RXNEIE	ERRIE	FRF	NSSP	SSOE	TXDMAEN	RXDMAEN	
	Reset value		0	0	0	0	1	1	1	0	0	0	0	0	0	0	
0x08	SPIx_SR	Res.	Res.	Res.	FTLVL[1:0]		FRLVL[1:0]		FRE	BSY	OVR	MODF	CRERR	UDR	CHSIDE	TXE	RXNE
	Reset value				0	0	0	0	0	0	0	0	0	0	0	1	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_CRCPDR	CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPIx_RXCRDR	RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPIx_TXCRDR	TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPIx_I2SCFGR	Res.	Res.	Res.	ASTRTEN	I2SMOD	I2SE	I2SCFG[1:0]		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN[1:0]		CHLEN
	Reset value				0	0	0	0	0	0		0	0	0	0	0	0
0x20	SPIx_I2SPR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
	Reset value							0	0	0	0	0	0	0	0	1	0

Refer to Section 2.2 on page 60 for the register boundary addresses.

## 36 FD controller area network (FDCAN)

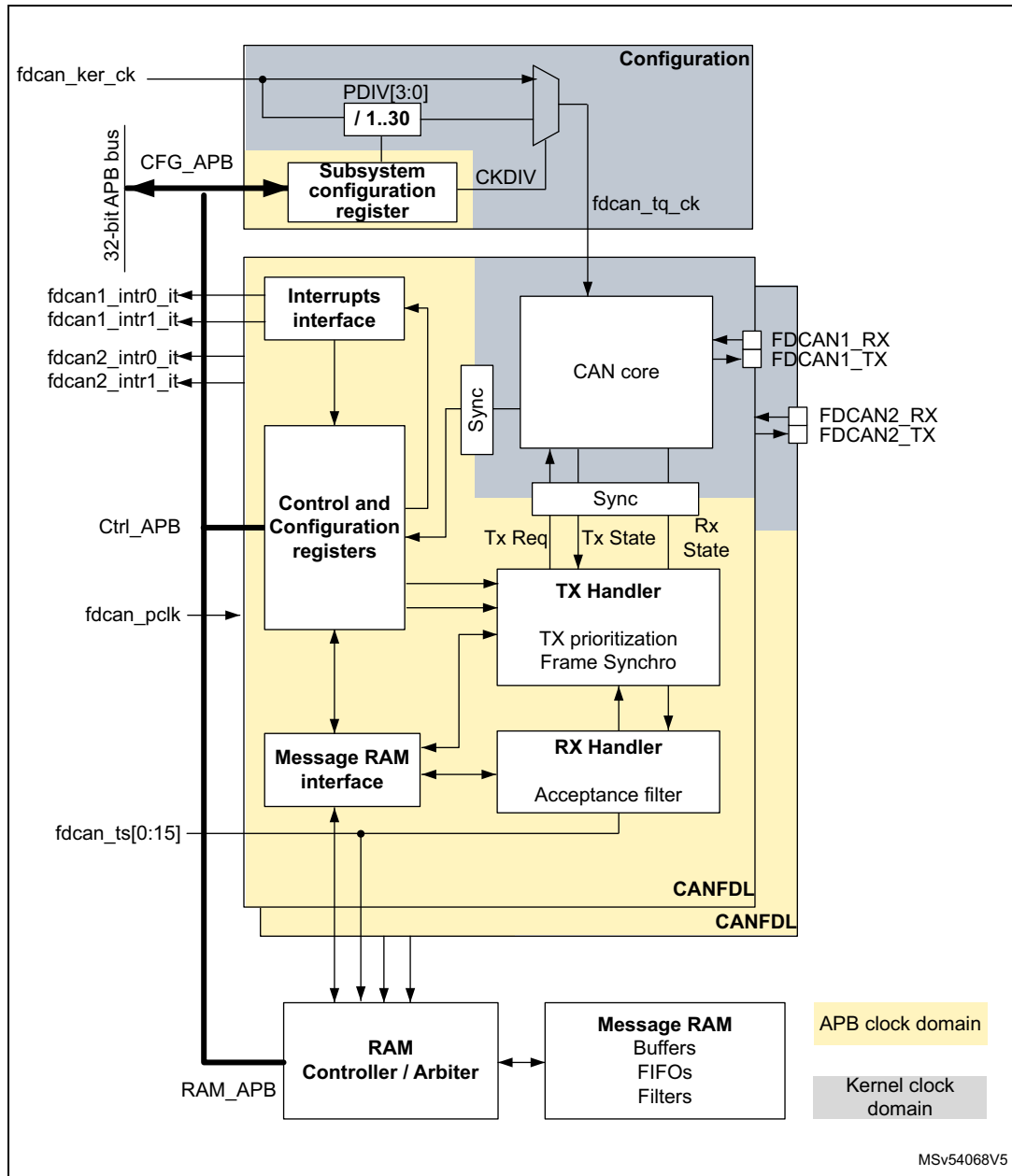
### 36.1 Introduction

The controller area network (CAN) subsystem (see [Figure 392](#)) consists of two CAN modules, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance is used for filtering, transmitting event FIFOs, and receiving and transmitting FIFOs.

Figure 392. CAN subsystem.



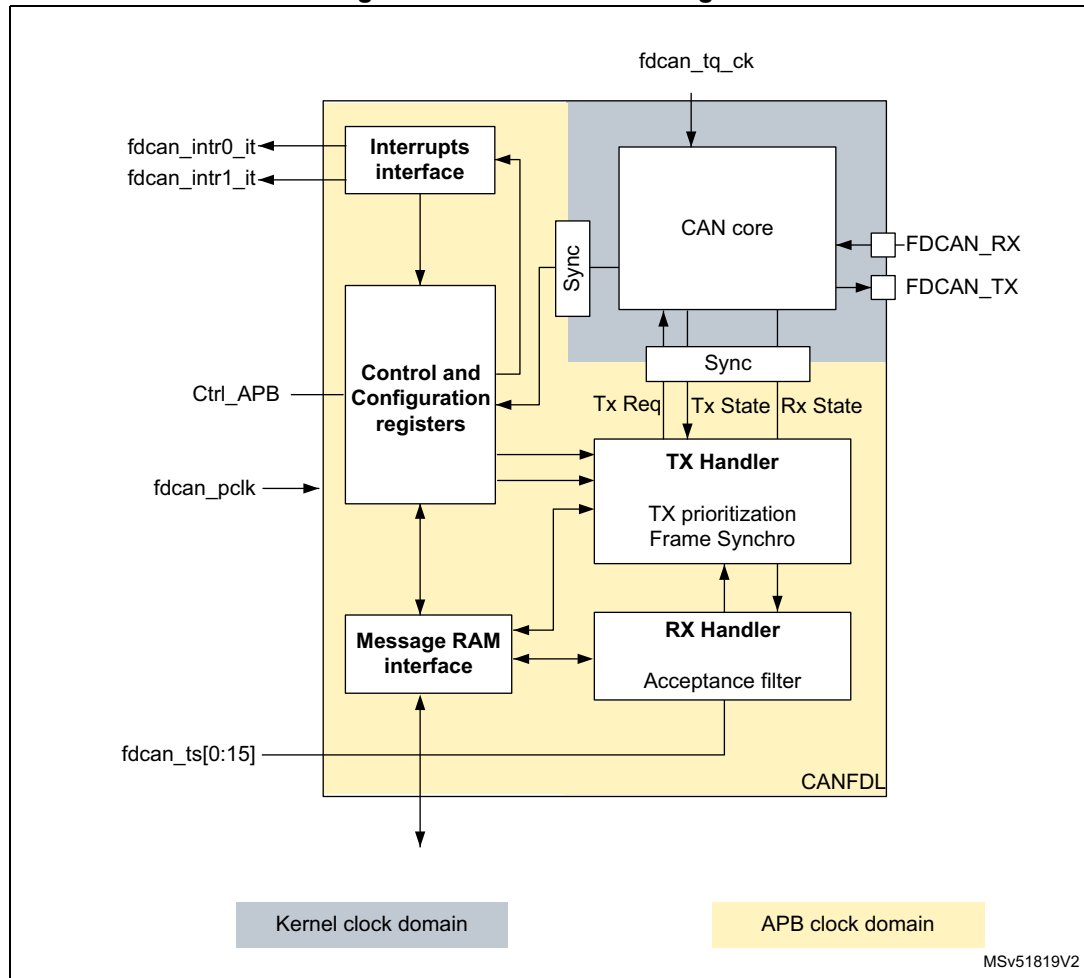
## 36.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

### 36.3 FDCAN functional description

#### 36.3.1 FDCAN block diagram

Figure 393. FDCAN block diagram



#### Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, `fdcan_intr0_it` and `fdcan_intr1_it`.

By programming the `EINT0` and `EINT1` bits of the `FDCAN_ILE` register, the interrupt lines can be independently enabled or disabled.

#### CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

#### Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

**Tx handler**

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

**Rx handler**

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

**APB interface**

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

**Message RAM interface**

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbitrator.

**36.3.2 FDCAN pins and internal signals**

The CAN subsystem I/O signals and pins are detailed, respectively, in [Table 210](#), [Table 211](#), and [Figure 392](#).

**Table 210. CAN subsystem I/O signals**

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcann_intr0_it	Digital output	FDCAN interrupt0
fdcann_intr1_it		FDCAN interrupt1
fdcan_ts[0:15]	-	External timestamp vector
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

**Table 211. CAN subsystem I/O pins**

Name	Type	Description
FDCAN <sub>n</sub> _RX	Digital input	FDCAN receive pin
FDCAN <sub>n</sub> _TX	Digital output	FDCAN transmit pin

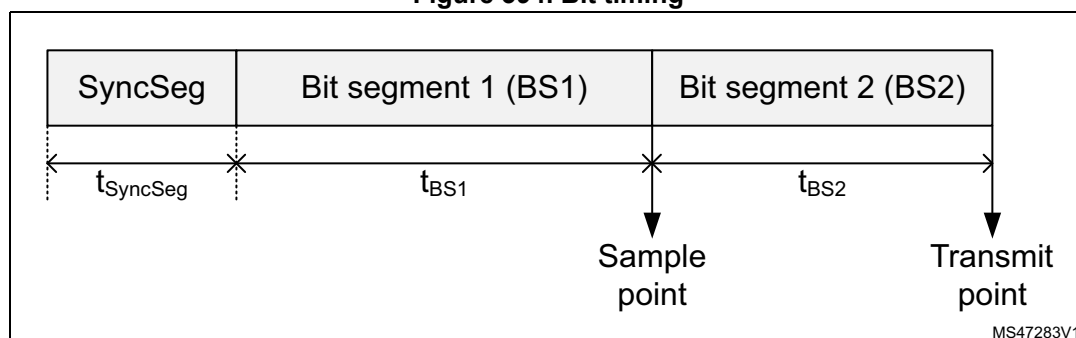
### 36.3.3 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in [Figure 394](#), this operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC\_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum ( $1 \times t_q$ ).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP\_SEG and PHASE\_SEG1 of the CAN standard. Its duration is programmable from 1 to 16 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE\_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but can also be automatically shortened to compensate for negative phase drifts.

Figure 394. Bit timing



The baud rate is the inverse of the bit time (baud rate =  $1 / \text{bit time}$ ), which, in turn, is the sum of three components (see [Figure 394](#)):

$$\text{bit time} = t_{\text{SyncSeg}} + t_{\text{BS1}} + t_{\text{BS2}}$$

Where:

- For the nominal bit time

$$t_q = (\text{NBRP}[8:0] + 1) \times t_{\text{fdcan\_tq\_clk}}$$

$$t_{\text{SyncSeg}} = 1 \times t_q$$

$$t_{\text{BS1}} = t_q \times (\text{NTSEG1}[7:0] + 1)$$

$$t_{\text{BS2}} = t_q \times (\text{NTSEG2}[6:0] + 1)$$

Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN\_NBTP register.

- For the data bit time

$$t_q = (\text{DBRP}[4:0] + 1) \times t_{\text{fdcan\_tq\_clk}}$$

$$t_{\text{SyncSeg}} = 1 \times t_q$$

$$t_{\text{BS1}} = t_q \times (\text{DTSEG1}[4:0] + 1)$$

$$t_{\text{BS2}} = t_q \times (\text{DTSEG2}[3:0] + 1)$$

Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN\_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC\_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC\_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN\_DBTP and FDCAN\_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA\_CCCR register are set.

The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock ( $f_{\text{dcan\_tq\_ck}} < f_{\text{dcan\_pclk}}$ ).

*Note:* For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

### 36.3.4 Operating modes

#### Configuration

Access to peripheral version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

#### Software initialization

Software initialization is started by setting the INIT bit of the FDCAN\_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN\_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN\_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN\_CCCR register are both set.

The CCE bit of the FDCAN\_CCCR register can only be set/cleared while the INIT bit of FDCAN\_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.



The following registers are reset when the CCE bit of the FDCAN\_CCCR register is set:

- FDCAN\_HPMS: High priority message status
- FDCAN\_RXF0S: Rx FIFO 0 status
- FDCAN\_RXF1S: Rx FIFO 1 status
- FDCAN\_TXFQS: Tx FIFO/queue status
- FDCAN\_TXBRP: Tx buffer request pending
- FDCAN\_TXBTO: Tx buffer transmission occurred
- FDCAN\_TXBCF: Tx buffer cancellation finished
- FDCAN\_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN\_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN\_TOCC register when the CCE bit of the FDCAN\_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN\_TXBAR: Tx buffer add request
- FDCAN\_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN\_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN\_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN\_CCCR can only be set/cleared while the INIT and CCE bits are both set.

### Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN\_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

### CAN FD operation

There are two variants in the FDCAN protocol:

- Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes.
- Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame.

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN\_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN\_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN\_PSR) to integrating (ACT[1:0] = 00 in FDCAN\_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN\_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN\_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN\_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRS bits set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to [Table 212](#).

**Table 212. DLC coding in FDCAN**

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN\_DBTP register. In the following

FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN\_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta ( $t_q$ ), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

### Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN\_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN\_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN\_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN\_TX through the transceiver to the receive input pin FDCAN\_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN\_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of  $mt_q$  (minimum time quantum, one period of `fdcan_tq_ck` clock).

The TDCV[6:0] bitfield of the FDCAN\_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN\_CCCR. It is

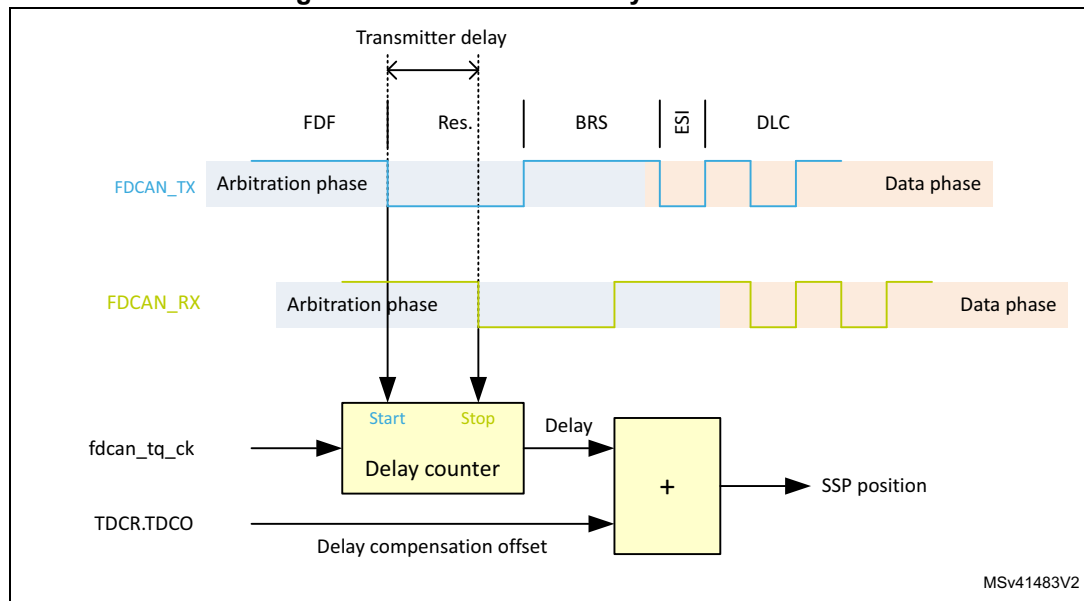
updated at each transmission of an FD frame while the TDC bit of the FDCAN\_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN\_Tx to FDCAN\_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN\_TX to FDCAN\_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to  $127 \times mt_q$ . If the sum exceeds this value, the maximum value ( $127 \times mt_q$ ) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN\_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN\_TX of the transmitter. The resolution of this measurement is one  $mt_q$ .

Figure 395. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN\_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN\_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN\_RX is low.

### Restricted operation mode

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN\_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN\_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN\_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN\_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

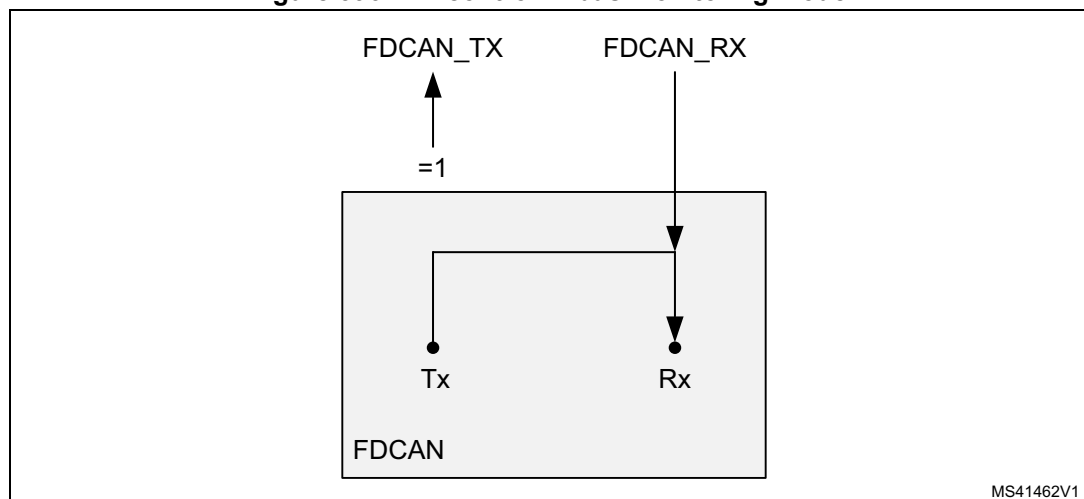
*Note: The restricted operation mode must not be combined with the loop-back mode (internal or external).*

### Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN\_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN\_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. [Figure 396](#) shows the connection of FDCAN\_TX and FDCAN\_RX signals to the FDCAN in bus monitoring mode.

**Figure 396. Pin control in bus monitoring mode**



MS41462V1

## Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN\_CCCR register.

### Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN\_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN\_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN\_TXBCF register.
- Successful transmission in spite of cancellation
  - The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN\_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN\_TXBCF register.
- Arbitration loss or frame transmission disturbed
  - The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN\_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN\_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

## Power-down (Sleep mode)

### Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN\_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN\_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN\_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

### Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

### Test modes

To enable write access to *FDCAN test register (FDCAN\_TEST)*, the TEST bit of the FDCAN\_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN\_TX by programming the TX[1:0] bitfield of the FDCAN\_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN\_RX can be read from the RX bit of FDCAN\_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN\_TX output pin. This applies also when reading FDCAN\_RX input pin via RX.

*Note:* *Test modes must be used for production tests or self-test only. The software control for FDCAN\_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.*

### External loop-back mode

The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN\_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 397* shows the connection of transmit and receive signals FDCAN\_TX and FDCAN\_RX to the FDCAN in external loop-back mode.

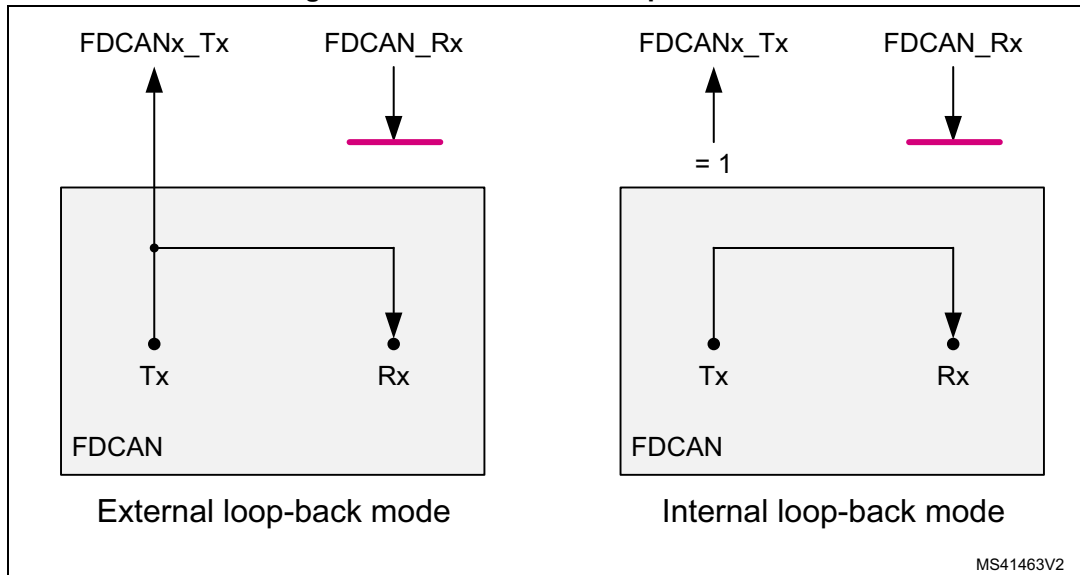
This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN\_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN\_TX transmit pin.

### Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN\_TEST and the MON bit of FDCAN\_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN\_TX and FDCAN\_RX pins. In this mode, FDCAN\_RX pin is disconnected from the FDCAN and FDCAN\_TX pin is held recessive. *Figure 397* shows the connection of FDCAN\_TX and FDCAN\_RX pins to the FDCAN in case of internal loop-back mode.



Figure 397. Pin control in loop-back mode



**Timestamp generation**

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN\_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN\_TSCV register. A write access to TSCV[15:0] resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN\_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming TSS[1:0] of FDCAN\_TSCC, a 16-bit timestamp can be used.

**Debug mode behavior**

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation

**Timeout counter**

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN\_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN\_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN\_TOCV. The timeout counter can only be started while the INIT bit of FDCAN\_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN\_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN\_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN\_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN\_TOCV has no effect.



When the counter reaches 0, the TOO interrupt flag is set in the FDCAN\_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

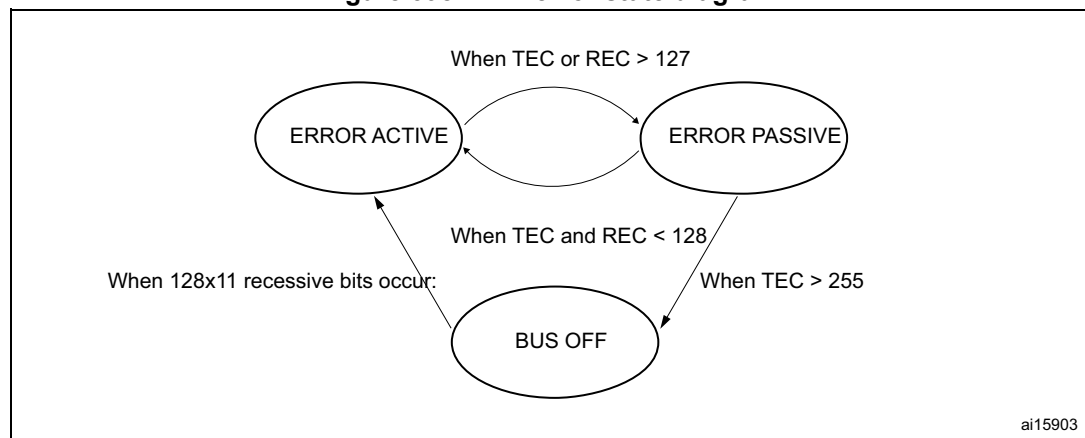
*Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.*

### 36.3.5 Error management

As described in the CAN protocol, the error management is handled entirely by hardware using the transmit error counter (the TEC[7:0] bitfield of the *FDCAN error counter register (FDCAN\_ECR)*) and the receive error counter (the REC[6:0] bitfield of the *FDCAN error counter register (FDCAN\_ECR)*). These values are incremented or decremented according to the error condition. For detailed information on TEC[7:0] and REC[6:0] management, refer to the CAN standard. Both values can be read by software to determine the stability of the network.

The bus-off state is reached when TEC[7:0] is greater than 255. This state is also indicated by the BO flag of the *FDCAN protocol status register (FDCAN\_PSR)*. In bus-off state, the CAN is no longer able to transmit and receive messages. It has to wait at least for the duration of the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on FDCAN\_RX input).

**Figure 398. CAN error state diagram**

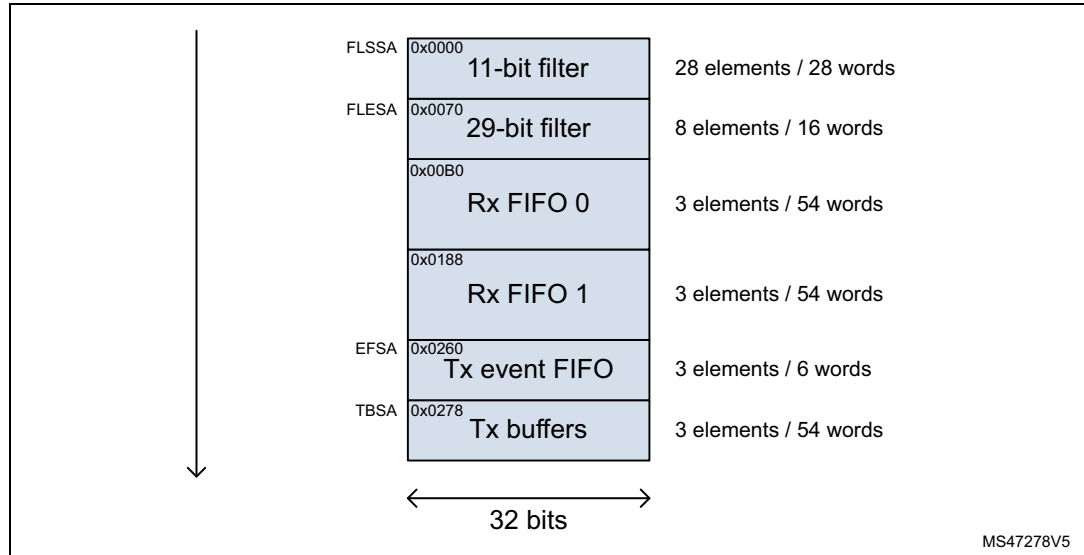


*Note: In initialization mode, the CAN does not monitor the FDCAN\_RX signal, and therefore cannot complete the recovery sequence. To recover from an error state, the CAN must operate in normal mode.*

### 36.3.6 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in [Figure 399](#).

**Figure 399. Message RAM configuration**



When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in [Figure 399](#))
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

#### Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

#### Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN\_IR
- Set the high priority message interrupt flag HPM in FDCAN\_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**  
The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN\_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in [Rx FIFO overwrite mode](#) have to be considered.

*Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.*

### Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT[1:0] = 00: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- EFT[1:0] = 11: the extended ID AND mask (XIDAM) is not used for range filtering.

### Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

### Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

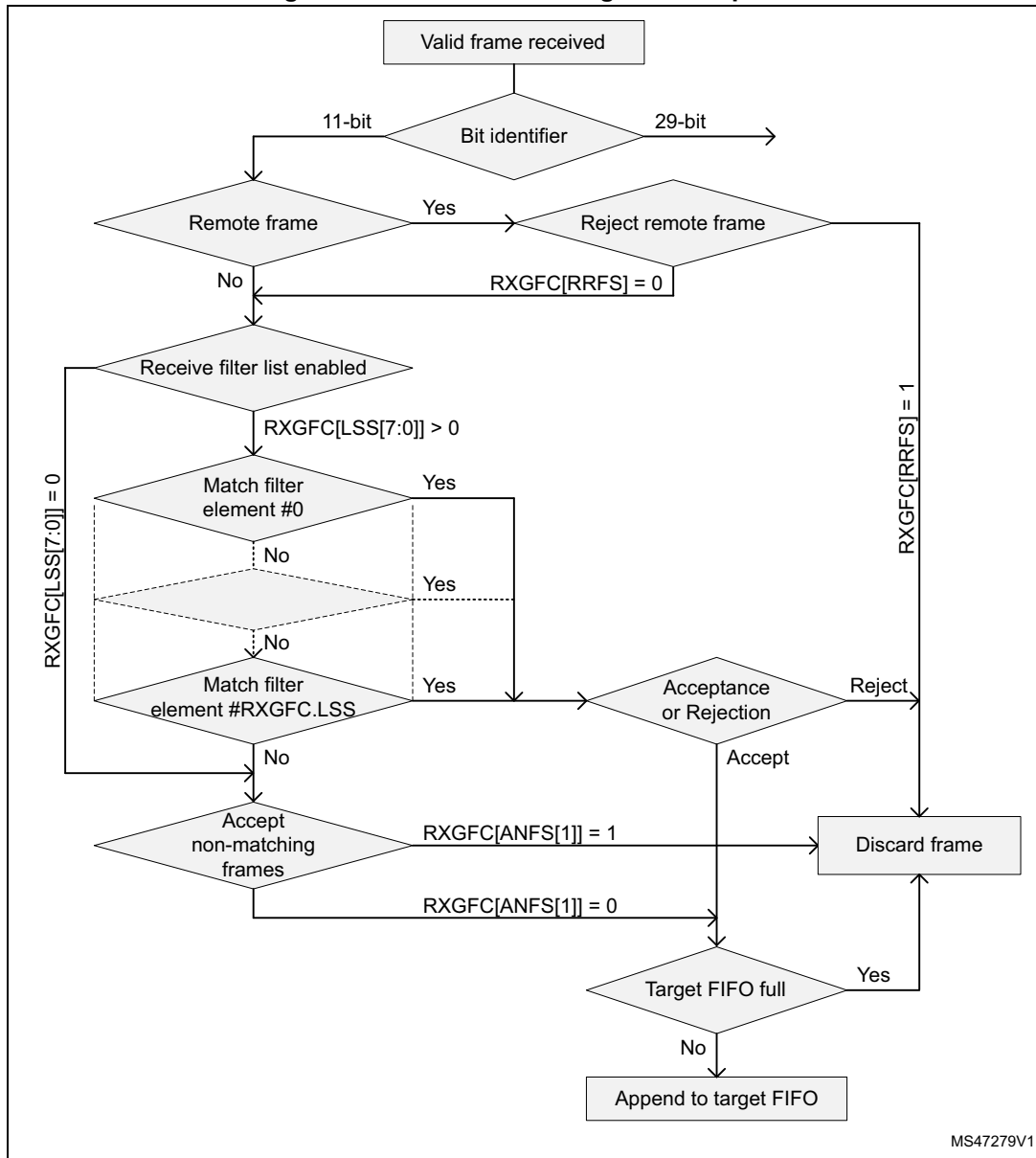
In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

### Standard message ID filtering

*Figure 400* shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in [Section 36.3.11](#).

The standard message filtering is controlled by the FDCAN\_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 400. Standard message ID filter path



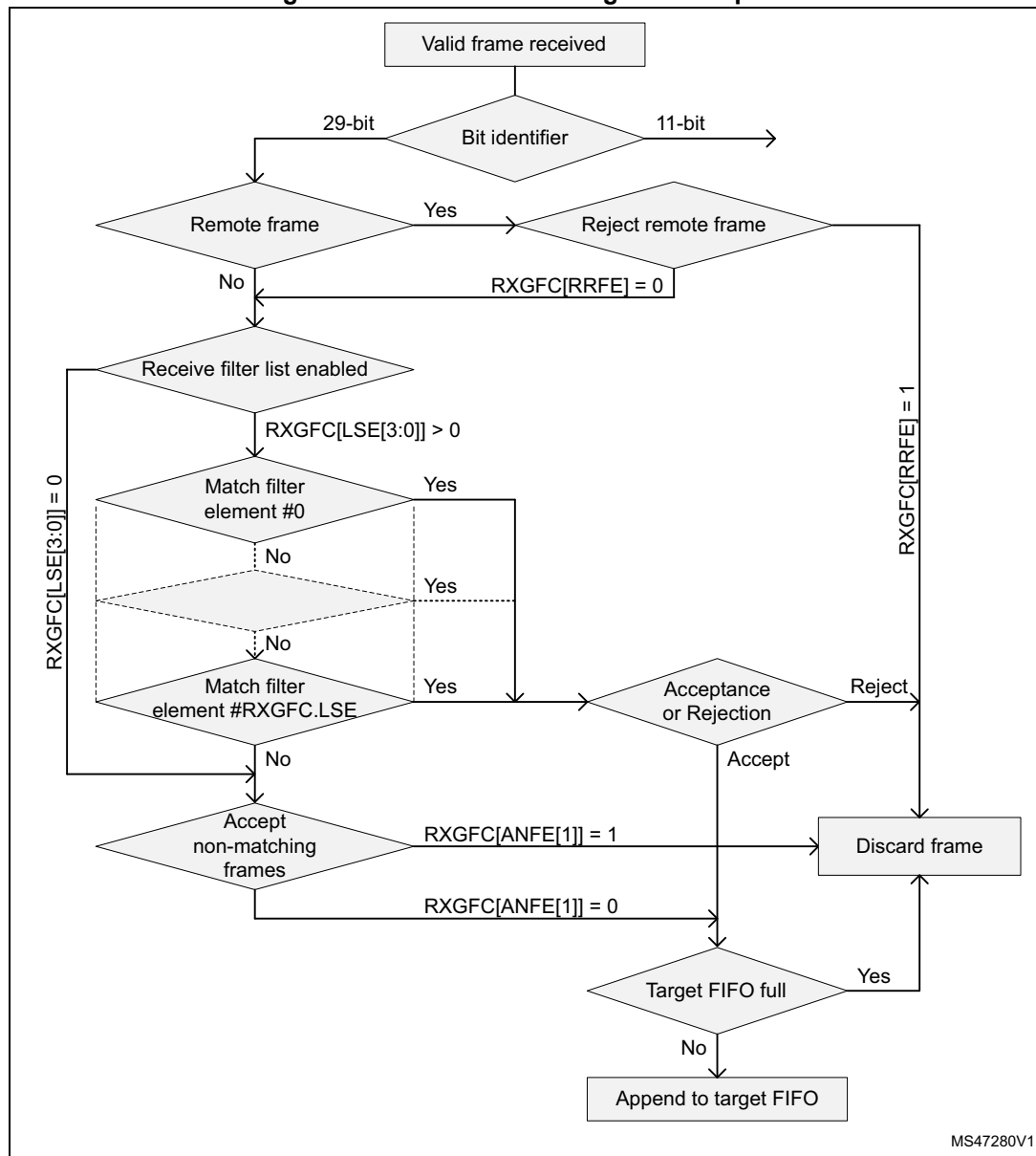
MS47279V1

**Extended message ID filtering**

Figure 401 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in Section 36.3.12.

The extended message filtering is controlled by the FDCAN\_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

**Figure 401. Extended message ID filter path**



The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

## Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see [Acceptance filter](#). The Rx FIFO element is described in [Section 36.3.8](#).

When an Rx FIFO full condition is signaled by RFnF in FDCAN\_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN\_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGI of FDCAN\_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

### Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN\_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN\_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN\_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN\_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN\_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN\_IR.

### Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN\_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGI of FDCAN\_RXFnS) is signaled by FnF = 1 in FDCAN\_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN\_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN\_RXFnS).

**Tx handling**

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. the Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

**Table 213. Possible configurations for frame transmission**

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

*Note:* AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN\_TXBRP) is updated, or when a transmission has been started.

**Transmit pause**

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN\_CCCR).



This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

### Tx FIFO

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN\_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN\_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN\_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN\_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN\_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN\_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCAN\_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding 18 times the put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

### Tx queue

Tx queue operation is configured by setting the TFQM of the FDCAN\_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN\_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN\_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application can use the FDCAN\_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding 18 times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

### Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN\_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN\_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN\_TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN\_TXBTO and FDCAN\_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN\_TXBCF bit is set.

*Note:* In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

### Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in [Tx FIFO](#).

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN\_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN\_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN\_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.

### 36.3.7 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see [Section 36.4.23](#) and [Section 36.4.25](#)). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

*Note:* The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

### 36.3.8 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in [Table 214](#). The description is provided in [Table 215](#).

**Table 214. Rx FIFO element**

Bit	31	24	23	16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]				
R1	ANMF	FIDX[6:0]		Res.	FDF	BRS	DLC[3:0]	RXTS[15:0]
R2	DB3[7:0]			DB2[7:0]			DB1[7:0]	D[7:0]
R3	DB7[7:0]			DB6[7:0]			DB5[7:0]	DB4[7:0]
⋮	⋮			⋮			⋮	
Rn	DBm[7:0]			DBm-1[7:0]			DBm-2[7:0]	DBm-3[7:0]

The element size configured for storage of CAN FD messages is set to 64-byte data field.

**Table 215. Rx FIFO element description**

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 215. Rx FIFO element description (continued)

Field	Description
R0 Bit 29 RTR	Remote transmission request Signals to the host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.
R2 Bits 31:24 DB3[7:0]	Data byte 3
R2 Bits 23:16 DB2[7:0]	Data byte 2
R2 Bits 15:8 DB1[7:0]	Data byte 1
R2 Bits 7:0 D[7:0]	Data byte 0
R3 Bits 31:24 DB7[7:0]	Data byte 7
R3 Bits 23:16 DB6[7:0]	Data byte 6

**Table 215. Rx FIFO element description (continued)**

Field	Description
R3 Bits 15:8 DB5[7:0]	Data byte 5
R3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data byte m
Rn Bits 23:16 DBm-1[7:0]	Data byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data byte m-3

### 36.3.9 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration TFQM bit of the FDCAN\_TXBC register. The element size is configured for storage of CAN FD messages with up to 64-byte data.

**Table 216. Tx buffer and FIFO element**

Bit	31	24	23	16	15	8	7	0	
T0	ESI	XTD	RTR	ID[28:0]					
T1	MM[7:0]			EFC	Res.	DFD	BRS	DLC[3:0]	Res.
T2	DB3[7:0]			DB2[7:0]			DB1[7:0]	D[7:0]	
T3	DB7[7:0]			DB6[7:0]			DB5[7:0]	DB4[7:0]	
⋮	⋮			⋮			⋮		
Tn	DBm[7:0]			DBm-1[7:0]			DBm-2[7:0]	DBm-3[7:0]	

**Table 217. Tx buffer element description**

Field	Description
T0 Bit 31 ESI <sup>(1)</sup>	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier

**Table 217. Tx buffer element description (continued)**

Field	Description
T0 Bit 29 RTR <sup>(2)</sup>	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS <sup>(3)</sup>	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data byte 3
T2 Bits 23:16 DB2[7:0]	Data byte 2
T2 Bits 15:8 DB1[7:0]	Data byte 1
T2 Bits 7:0 D[7:0]	Data byte 0
T3 Bits 31:24 DB7[7:0]	Data byte 7
T3 Bits 23:16 DB6[7:0]	Data byte 6
T3 Bits 15:8 DB5[7:0]	Data byte 5
T3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮

**Table 217. Tx buffer element description (continued)**

Field	Description
Tn Bits 31:24 DBm[7:0]	Data byte m
Tn Bits 23:16 DBm-1[7:0]	Data byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node can optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if the transmission in CAN FD format is enabled by the FDOE bit of the FDCAN\_CCCR.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled by setting the FDOE bit of the FDCAN\_CCCR. Bit BRS is only evaluated when in addition BRSE bit is set in FDCAN\_CCCR.

### 36.3.10 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN\_TXEFS register.

**Table 218. Tx event FIFO element**

Bit	31	24	23	16	15	8	7	0
E0	ESI	XTD	RTR	ID[28:0]				
E1	MM[7:0]			ET[1:0]	EDL	BRS	DLC[3:0]	TXTS[15:0]

**Table 219. Tx event FIFO element description**

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

**Table 219. Tx event FIFO element description (continued)**

Field	Description
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.

**36.3.11 FDCAN standard message ID filter element**

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 ... 27).

**Table 220. Standard message ID filter element**

Bit	31	24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]	SFID1[10:0]			Res.	SFID2[10:0]	



**Table 221. Standard message ID filter element field description**

Field	Description
Bit 31:30 SFT[1:0] <sup>(1)</sup>	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT[1:0] = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC[2:0] = 000).

*Note:* In case a reserved value is configured, the filter element is considered disabled.

### 36.3.12 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 ... 7).

**Table 222. Extended message ID filter element**

Bit	31	24	23	16	15	8	7	0
F0	EFEC[2:0]		EFID1[28:0]					
F1	EFT[1:0]	Res.	EFID2[28:0]					

**Table 223. Extended message ID filter element field description**

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

### 36.4 FDCAN registers

#### 36.4.1 FDCAN core release register (FDCAN\_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
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
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

#### 36.4.2 FDCAN endian register (FDCAN\_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

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

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

*Note:* The register read must give the reset value to ensure no endianness issue.

#### 36.4.3 FDCAN data bit timing and prescaler register (FDCAN\_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN\_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods:  
 $t_q = (\text{DBRP}[4:0] + 1) \text{ FDCAN clock periods.}$

DTSEG1[4:0] is the sum of PROP\_SEG and PHASE\_SEG1. DTSEG2[3:0] is PHASE\_SEG2. Therefore, the length of the bit time is  
 (programmed values)  $\times [DTSEG1[4:0] + DTSEG2[3:0] + 3] \times t_q$  or  
 (functional values)  $\times [SYNC\_SEG + PROP\_SEG + PHASE\_SEG1 + PHASE\_SEG2] \times t_q$ .

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								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.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			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 **TDC**: Transceiver delay compensation  
 0: Transceiver delay compensation disabled  
 1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler  
 The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values for the baud rate prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point  
 Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is  $t_{BS1} = (DTSEG1[4:0] + 1) \times t_q$ .

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point  
 Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e.  $t_{BS2} = (DTSEG2[3:0] + 1) \times t_q$ .

Bits 3:0 **DSJW[3:0]**: Synchronization jump width  
 Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1:  $t_{SJW} = (DSJW[3:0] + 1) \times t_q$ .

*Note:* With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s.  
 The data phase bit rate must be higher than or equal to the nominal bit rate.

### 36.4.4 FDCAN test register (FDCAN\_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN\_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx\_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010

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.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

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

Bit 7 **RX**: Receive pin

This bit is used to monitor the actual value of FDCANx\_RX. It is synchronized with the FDCANx\_RX pin, so it is set after reset if the FDCAN is connected to a network.

0: The CAN bus is dominant (FDCANx\_RX = 0)

1: The CAN bus is recessive (FDCANx\_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx\_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx\_TX

10: Dominant (0) level at pin FDCANx\_TX

11: Recessive (1) at pin FDCANx\_TX

Bit 4 **LBCK**: Loop-back mode

0: Reset value, loop-back mode is disabled

1: Loop-back mode is enabled (see [Power-down \(Sleep mode\)](#))

Bits 3:0 Reserved, must be kept at reset value.

### 36.4.5 FDCAN RAM watchdog register (FDCAN\_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN\_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN\_IR register. The RAM watchdog counter is clocked by the fdcan\_pclk clock.

Address offset: 0x0014

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
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:8 **WDV[7:0]**: Watchdog value  
 Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration  
 Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.  
 This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

### 36.4.6 FDCAN CC control register (FDCAN\_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see [Software initialization](#).

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
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rW	rW	rW	rW			rW	rW	rW	rW	rW	rW	r	rW	rW	rW

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

Bit 15 **NISO**: Non-ISO operation  
 If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.  
 0: CAN FD frame format according to ISO11898-1  
 1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP**: Transmit pause enable  
 If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.  
 0: Disabled  
 1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration  
 0: Edge filtering disabled  
 1: Two consecutive dominant  $t_q$  required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable  
 0: Protocol exception handling enabled  
 1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching  
 0: Bit rate switching for transmissions disabled  
 1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable  
 0: FD operation disabled  
 1: FD operation enabled

- Bit 7 **TEST**: Test mode enable  
 0: Normal operation, FDCAN\_TEST holds reset values  
 1: Test mode, write access to FDCAN\_TEST enabled
- Bit 6 **DAR**: Disable automatic retransmission  
 0: Automatic retransmission of messages not transmitted successfully enabled  
 1: Automatic retransmission disabled
- Bit 5 **MON**: Bus monitoring mode  
 This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.  
 0: Bus monitoring mode disabled  
 1: Bus monitoring mode enabled
- Bit 4 **CSR**: Clock stop request  
 0: No clock stop requested  
 1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.
- Bit 3 **CSA**: Clock stop acknowledge  
 0: No clock stop acknowledged  
 1: FDCAN can be set in power-down by stopping APB clock and kernel clock.
- Bit 2 **ASM**: ASM restricted operation mode  
 The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.  
 0: Normal CAN operation  
 1: Restricted operation mode active
- Bit 1 **CCE**: Configuration change enable  
 0: The CPU has no write access to the protected configuration registers.  
 1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN\_CCCR).
- Bit 0 **INIT**: Initialization  
 0: Normal operation  
 1: Initialization started

*Note:* Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

### 36.4.7 FDCAN nominal bit timing and prescaler register (FDCAN\_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN\_CCCR register are both set. The CAN bit time can be programmed in the range of  $4$  to  $81 \times t_q$ . The CAN time quantum can be programmed in the range of  $1$  to  $1024$  FDCAN kernel clock periods:  
 $t_q = (BRP + 1) \times \text{FDCAN clock period } fdcan\_ker\_ck$ .

NTSEG1[7:0] is the sum of PROP\_SEG and PHASE\_SEG1. NTSEG2[6:0] is PHASE\_SEG2. Therefore, the length of the bit time is  
 (programmed values)  $\times [NTSEG1[7:0] + NTSEG2[6:0] + 3] \times t_q$  or  
 (functional values)  $\times [SYNC\_SEG + PROP\_SEG + PHASE\_SEG1 + PHASE\_SEG2] \times t_q$ .

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]								NBRP[8:0]							
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width  
 Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.  
 This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler  
 Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.  
 This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point  
 Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.  
 This bitfield is write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point  
 Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

*Note:* With a CAN kernel clock of 48 MHz, the reset value of 0x0600 0A03 configures the FDCAN for a bit rate of 3 Mbit/s.



### 36.4.8 FDCAN timestamp counter configuration register (FDCAN\_TSCC)

Address offset: 0x0020

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.	TCP[3:0]			
												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.	TSS[1:0]	
														rw	rw

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

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1...16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

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

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value (tim3\_cnt[0:15])

11: Same as 00.

These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

### 36.4.9 FDCAN timestamp counter value register (FDCAN\_TSCV)

Address offset: 0x0024

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
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

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

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSS[1:0] = 01 in FDCAN\_TSCC, the timestamp counter is incremented in multiples of CAN bit times [1 ... 16] depending on the configuration of TCP[3:0] in FDCAN\_TSCC. A wrap around sets the TSW interrupt flag in FDCAN\_IR. Write access resets the counter to 0. When TSS[1:0] = 10, TSC[15:0] reflects the external timestamp counter value. A write access has no impact.

*Note:* A “wrap around” is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN\_TSCV.

### 36.4.10 FDCAN timeout counter configuration register (FDCAN\_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

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

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period. This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in continuous mode, a write to FDCAN\_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN\_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.

- 00: Continuous operation
- 01: Timeout controlled by Tx event FIFO
- 10: Timeout controlled by Rx FIFO 0
- 11: Timeout controlled by Rx FIFO 1

This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable

- 0: Timeout counter disabled
- 1: Timeout counter enabled

This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

For more details, see [Timeout counter](#).

### 36.4.11 FDCAN timeout counter value register (FDCAN\_TOCV)

Address offset: 0x002C

Reset value: 0x0000 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
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

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

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 ... 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN\_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN\_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN\_TOCC.

### 36.4.12 FDCAN error counter register (FDCAN\_ECR)

Address offset: 0x0040

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.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN\_IR. Access type is rc\_r: cleared on read.

Bit 15 **RP**: Receive error passive

- 0: The receive error counter is below the error passive level of 128.
- 1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]**: Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When the ASM bit of the FDCAN\_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented.

### 36.4.13 FDCAN protocol status register (FDCAN\_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

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.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBR	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN\_TX to FDCAN\_RX and TDCO[6:0] in FDCAN\_TDCR. The SSP position is, in the data phase, the number of minimum time quanta ( $mt_q$ ) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to  $127 \times mt_q$ .

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

- 0: No protocol exception event occurred since last read access
- 1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.  
 0: Since this bit was cleared by the CPU, no FDCAN message has been received.  
 1: Message in FDCAN format with EDL flag set has been received.  
 Access type is rc\_r: cleared on read.

Bit 12 **RBR**: BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.  
 0: Last received FDCAN message did not have its BRS flag set.  
 1: Last received FDCAN message had its BRS flag set.  
 Access type is rc\_r: cleared on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.  
 0: Last received FDCAN message did not have its ESI flag set.  
 1: Last received FDCAN message had its ESI flag set.  
 Access type is rc\_r: cleared on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.  
 Access type is rs: set on read.

Bit 7 **BO**: Bus-off status

- 0: The FDCAN is not in bus-off state.
- 1: The FDCAN is in bus-off state.

Bit 6 **EW**: Warning status

0: Both error counters are below the error-warning limit of 96.

1: At least one of error counter has reached the error-warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the error-passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]**: Last error code

LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.

000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.

001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error. A fixed format part of a received frame has the wrong format.

011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is rs: set on read.

**Note:** *When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.*

*The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN\_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN\_PSR, enabling the CPU to check up whether the CAN bus is*

stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The REC[6:0] bitfield of the FDCAN\_ECR register is used to count these sequences.

### 36.4.14 FDCAN transmitter delay compensation register (FDCAN\_TDCR)

Address offset: 0x0048

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.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN\_TX to FDCAN\_RX and the secondary sample point. Valid values are 0 to 127 × mt<sub>q</sub>.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN\_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

### 36.4.15 FDCAN interrupt register (FDCAN\_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN\_IE controls whether an interrupt is generated. The configuration of FDCAN\_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

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.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
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 31:24 Reserved, must be kept at reset value.

- Bit 23 **ARA**: Access to reserved address  
0: No access to reserved address occurred  
1: Access to reserved address occurred
- Bit 22 **PED**: Protocol error in data phase (data bit time is used)  
0: No protocol error in data phase  
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN\_PSR)
- Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)  
0: No protocol error in arbitration phase  
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN\_PSR)
- Bit 20 **WDI**: Watchdog interrupt  
0: No message RAM watchdog event occurred  
1: Message RAM watchdog event due to missing READY
- Bit 19 **BO**: Bus-off status  
0: Bus-off status unchanged  
1: Bus-off status changed
- Bit 18 **EW**: Warning status  
0: Error-warning status unchanged  
1: Error-warning status changed
- Bit 17 **EP**: Error passive  
0: Error-passive status unchanged  
1: Error-passive status changed
- Bit 16 **ELO**: Error logging overflow  
0: CAN error logging counter did not overflow.  
1: Overflow of CAN error logging counter occurred.
- Bit 15 **TOO**: Timeout occurred  
0: No timeout  
1: Timeout reached
- Bit 14 **MRAF**: Message RAM access failure  
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
  - was unable to write a message to the message RAM. In this case message storage is aborted.
- In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.  
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see [Restricted operation mode](#)). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN\_CCCR register.
- 0: No message RAM access failure occurred  
1: Message RAM access failure occurred

- Bit 13 **TSW**: Timestamp wraparound  
0: No timestamp counter wrap-around  
1: Timestamp counter wrapped around
- Bit 12 **TEFL**: Tx event FIFO element lost  
0: No Tx event FIFO element lost  
1: Tx event FIFO element lost
- Bit 11 **TEFF**: Tx event FIFO full  
0: Tx event FIFO Not full  
1: Tx event FIFO full
- Bit 10 **TEFN**: Tx event FIFO new entry  
0: Tx event FIFO unchanged  
1: Tx handler wrote Tx event FIFO element.
- Bit 9 **TFE**: Tx FIFO empty  
0: Tx FIFO non-empty  
1: Tx FIFO empty
- Bit 8 **TCF**: Transmission cancellation finished  
0: No transmission cancellation finished  
1: Transmission cancellation finished
- Bit 7 **TC**: Transmission completed  
0: No transmission completed  
1: Transmission completed
- Bit 6 **HPM**: High-priority message  
0: No high-priority message received  
1: High-priority message received
- Bit 5 **RF1L**: Rx FIFO 1 message lost  
0: No Rx FIFO 1 message lost  
1: Rx FIFO 1 message lost
- Bit 4 **RF1F**: Rx FIFO 1 full  
0: Rx FIFO 1 not full  
1: Rx FIFO 1 full
- Bit 3 **RF1N**: Rx FIFO 1 new message  
0: No new message written to Rx FIFO 1  
1: New message written to Rx FIFO 1
- Bit 2 **RF0L**: Rx FIFO 0 message lost  
0: No Rx FIFO 0 message lost  
1: Rx FIFO 0 message lost
- Bit 1 **RF0F**: Rx FIFO 0 full  
0: Rx FIFO 0 not full  
1: Rx FIFO 0 full
- Bit 0 **RF0N**: Rx FIFO 0 new message  
0: No new message written to Rx FIFO 0  
1: New message written to Rx FIFO 0



### 36.4.16 FDCAN interrupt enable register (FDCAN\_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

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.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

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

Bit 23 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable

0: Interrupt disabled

1: Interrupt enabled

Bit 19 **BOE**: Bus-off status

0: Interrupt disabled

1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable

0: Interrupt disabled

1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable

0: Interrupt disabled

1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable

0: Interrupt disabled

1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable

0: Interrupt disabled

1: Interrupt enabled

Bit 14 **MRAFE**: Message RAM access failure interrupt enable

0: Interrupt disabled

1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable

0: Interrupt disabled

1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable

0: Interrupt disabled

1: Interrupt enabled

- Bit 11 **TEFFE**: Tx event FIFO full interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 9 **TFFEE**: Tx FIFO empty interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 8 **TCFE**: Transmission cancellation finished interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 7 **TCE**: Transmission completed interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 6 **HPME**: High-priority message interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 2 **RFOLE**: Rx FIFO 0 message lost interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 1 **RFOFE**: Rx FIFO 0 full interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled
- Bit 0 **RFONE**: Rx FIFO 0 new message interrupt enable  
0: Interrupt disabled  
1: Interrupt enabled

### 36.4.17 FDCAN interrupt line select register (FDCAN\_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line has to be enabled via the EINT0 and EINT1 bit of the FDCAN\_ILE register.

Address offset: 0x0058

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.	PERR	BERR	MISC	TFERR	SMSG	RXFIF O1	RXFIF O0
									r/w	r/w	r/w	r/w	r/w	r/w	r/w

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

Bit 6 **PERR**: Protocol error grouping the following interruption

- ARAL: Access to reserved address line
- PEDL: Protocol error in data phase line
- PEAL: Protocol error in arbitration phase line
- WDIL: Watchdog interrupt line
- BOL: Bus-off status
- EWL: Warning status interrupt line

Bit 5 **BERR**: Bit and line error grouping the following interruption

- EPL Error passive interrupt line
- ELOL: Error logging overflow interrupt line

Bit 4 **MISC**: Interrupt regrouping the following interruption

- TOOL: Timeout occurred interrupt line
- MRAFL: Message RAM access failure interrupt line
- TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR**: Tx FIFO ERROR grouping the following interruption

- TEFLL: Tx event FIFO element lost interrupt line
- TEFFL: Tx event FIFO full interrupt line
- TEFNL: Tx event FIFO new entry interrupt line
- TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG**: Status message bit grouping the following interruption

- TCFL: Transmission cancellation finished interrupt line
- TCL: Transmission completed interrupt line
- HPML: High-priority message interrupt line

Bit 1 **RXFIFO1**: RX FIFO bit grouping the following interruption

- RF1LL: Rx FIFO 1 message lost interrupt line
- RF1FL: Rx FIFO 1 full interrupt line
- RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RXFIFO0**: RX FIFO bit grouping the following interruption  
 RF0LL: Rx FIFO 0 message lost interrupt line  
 RF0FL: Rx FIFO 0 full interrupt line  
 RF0NL: Rx FIFO 0 new message interrupt line

### 36.4.18 FDCAN interrupt line enable register (FDCAN\_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

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.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1  
 0: Interrupt line fdcan\_intr0\_it disabled  
 1: Interrupt line fdcan\_intr0\_it enabled

Bit 0 **EINT0**: Enable interrupt line 0  
 0: Interrupt line fdcan\_intr1\_it disabled  
 1: Interrupt line fdcan\_intr1\_it enabled

### 36.4.19 FDCAN global filter configuration register (FDCAN\_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in [Figure 400](#) and [Figure 401](#).

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

- Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list  
0: No extended message ID filter  
1 to 8: Number of extended message ID filter elements  
> 8: Values greater than 8 are interpreted as 8.  
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.
- Bits 23:21 Reserved, must be kept at reset value.
- Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list  
0: No standard message ID filter  
1 to 28: Number of standard message ID filter elements  
> 28: Values greater than 28 are interpreted as 28.  
This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.
- Bits 15:10 Reserved, must be kept at reset value.
- Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)  
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.
- Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)  
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.
- Bits 7:6 Reserved, must be kept at reset value.
- Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard  
Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.  
00: Accept in Rx FIFO 0  
01: Accept in Rx FIFO 1  
10: Reject  
11: Reject  
This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.
- Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended  
Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.  
00: Accept in Rx FIFO 0  
01: Accept in Rx FIFO 1  
10: Reject  
11: Reject  
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.
- Bit 1 **RRFS**: Reject remote frames standard  
0: Filter remote frames with 11-bit standard IDs  
1: Reject all remote frames with 11-bit standard IDs  
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bit 0 **RRFE**: Reject remote frames extended

0: Filter remote frames with 29-bit standard IDs

1: Reject all remote frames with 29-bit standard IDs

This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

### 36.4.20 FDCAN extended ID and mask register (FDCAN\_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask

For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

### 36.4.21 FDCAN high-priority message status register (FDCAN\_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

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
FLST	Res.	Res.	FIDX[4:0]				MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]			
r			r	r	r	r	r	r	r				r	r	r

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

Bit 15 **FLST**: Filter list

Indicates the filter list of the matching filter element:

0: Standard filter list

1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index

Index of matching filter element.  
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN\_RXGFC.

Bits 7:6 **MSI[1:0]**: Message storage indicator

00: No FIFO selected  
01: FIFO overrun  
10: Message stored in FIFO 0  
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index

Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

### 36.4.22 FDCAN Rx FIFO 0 status register (FDCAN\_RXF0S)

Address offset: 0x0090

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.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost

This bit is a copy of the RF0L interrupt flag of the FDCAN\_IR register. When RF0L is cleared, this bit is also cleared.

0: No Rx FIFO 0 message lost  
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full

0: Rx FIFO 0 not full  
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index

Rx FIFO 0 write index pointer.  
Range: 0 to 2.

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

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index

Rx FIFO 0 read index pointer.  
Range: 0 to 2.

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

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level

Number of elements stored in Rx FIFO 0.  
Range: 0 to 3.

### 36.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN\_RXF0A)

Address offset: 0x0094

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.	Res.	F0AI[2:0]		
													rw	rw	rw

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

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index

After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN\_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN\_RXF0S).

### 36.4.24 FDCAN Rx FIFO 1 status register (FDCAN\_RXF1S)

Address offset: 0x0098

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.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of the RF1L interrupt flag of the FDCAN\_IR register. When RF1L is cleared, this bit is also cleared.

0: No Rx FIFO 1 message lost

1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full

0: Rx FIFO 1 not full

1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer.

Range: 0 to 2.

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



Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index  
 Rx FIFO 1 read index pointer.  
 Range: 0 to 2.

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

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level  
 Number of elements stored in Rx FIFO 1.  
 Range: 0 to 3.

### 36.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN\_RXF1A)

Address offset: 0x009C

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.	Res.	F1AI[2:0]		
													rw	rw	rw

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

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index  
 After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN\_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN\_RXF1S).

### 36.4.26 FDCAN Tx buffer configuration register (FDCAN\_TXBC)

Address offset: 0x00C0

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.	TFQM	Res.	Res.	Res.	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

Bit 24 **TFQM**: Tx FIFO/queue mode  
 0: Tx FIFO operation  
 1: Tx queue operation.  
 This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bits 23:0 Reserved, must be kept at reset value.



### 36.4.27 FDCAN Tx FIFO/queue status register (FDCAN\_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN\_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN\_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0003

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.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										r				r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
						r	r						r	r	r

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

Bit 21 **TFQF**: Tx FIFO/queue full  
 0: Tx FIFO/queue not full  
 1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index  
 Tx FIFO/queue write index pointer, range 0 to 3

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

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index  
 Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN\_TXBC)

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

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level  
 Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN\_TXBC).

### 36.4.28 FDCAN Tx buffer request pending register (FDCAN\_TXBRP)

Address offset: 0x00C8

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.	Res.	TRP[2:0]		
													r	r	r

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

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN\_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN\_TXBCR register.

After the FDCAN\_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN\_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN\_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN\_TXBCF in the following cases:

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN\_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

*Note:* FDCAN\_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN\_TXBRP bit is cleared.

### 36.4.29 FDCAN Tx buffer add request register (FDCAN\_TXBAR)

Address offset: 0x00CC

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.	Res.	AR[2:0]		
													rw	rw	rw

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

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN\_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

*Note:* If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN\_TXBRP bit already set), the request is ignored.



### 36.4.30 FDCAN Tx buffer cancellation request register (FDCAN\_TXBCR)

Address offset: 0x00D0

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.	Res.	CR[2:0]		
													rw	rw	rw

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

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN\_TXBCR. The bits remain set until the corresponding FDCAN\_TXBRP bit is cleared.

- 0: No cancellation pending
- 1: Cancellation pending

### 36.4.31 FDCAN Tx buffer transmission occurred register (FDCAN\_TXBTO)

Address offset: 0x00D4

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.	Res.	TO[2:0]		
													r	r	r

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

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN\_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN\_TXBAR.

- 0: No transmission occurred
- 1: Transmission occurred

### 36.4.32 FDCAN Tx buffer cancellation finished register (FDCAN\_TXBCF)

Address offset: 0x00D8

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.	Res.	CF[2:0]		
													r	r	r

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

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN\_TXBRP bit is cleared after a cancellation was requested via FDCAN\_TXBCR. In case the corresponding FDCAN\_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN\_TXBAR register.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

### 36.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN\_TXBTIE)

Address offset: 0x00DC

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.	Res.	TIE[2:0]		
													rw	rw	rw

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

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

### 36.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN\_TXBCIE)

Address offset: 0x00E0

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.	Res.	CFIE[2:0]		
													rw	rw	rw

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

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.

Each Tx buffer has its own CFIE bit.

0: Cancellation finished interrupt disabled

1: Cancellation finished interrupt enabled

### 36.4.35 FDCAN Tx event FIFO status register (FDCAN\_TXEFS)

Address offset: 0x00E4

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.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EFGI[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
						r	r						r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost

This bit is a copy of the TEFL interrupt flag of the FDCAN\_IR. When TEFL is cleared, this bit is also cleared.

0 No Tx event FIFO element lost

1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full

0: Tx event FIFO not full

1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index

Tx event FIFO write index pointer.

Range: 0 to 3.

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

Bits 9:8 **EFGI[1:0]**: Event FIFO get index  
 Tx event FIFO read index pointer.  
 Range: 0 to 3.

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

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level  
 Number of elements stored in Tx event FIFO.  
 Range: 0 to 3.

### 36.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN\_TXEFA)

Address offset: 0x00E8

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.	Res.	Res.	EFAI[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index  
 After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN\_TXEFS) to EFAI[1:0] + 1 and updates the FIFO 0 fill level (EFFL[2:0] of FDCAN\_TXEFS).

### 36.4.37 FDCAN CFG clock divider register (FDCAN\_CKDIV)

Address offset: 0x0100

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.	PDIV[3:0]			
												rw	rw	rw	rw

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

Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

- 0000: Divide by 1
- 0001: Divide by 2
- 0010: Divide by 4
- 0011: Divide by 6
- 0100: Divide by 8
- 0101: Divide by 10
- 0110: Divide by 12
- 0111: Divide by 14
- 1000: Divide by 16
- 1001: Divide by 18
- 1010: Divide by 20
- 1011: Divide by 22
- 1100: Divide by 24
- 1101: Divide by 26
- 1110: Divide by 28
- 1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE bit of the FDCAN\_CCCR register is set.

*Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN\_CKDIV register, which changes clock divider for all instances.*

*If several FDCAN instances are present, the input clock divider must be modified before configuring the other FDCAN instances.*

### 36.4.38 FDCAN register map

**Table 224. FDCAN register map and reset values**

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]			STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			MON[7:0]				DAY[7:0]													
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	1	
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res	Res	Res	Res	Res	Res	Res	Res	TDC	Res	Res	DBRP[4:0]				Res	Res	Res	DTSEG1[4:0]				DTSEG2[3:0]			DSJW[3:0]						
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	0	1	1	0	0	1
0x0010	FDCAN_TEST	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RX	TX[1:0]		LBCK	Res	Res	Res	Res
	Reset value																									0	0	0	0				





Table 224. FDCAN 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				
0x0014	FDCAN_RWD	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WDV[7:0]							WDC[7:0]												
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x0018	FDCAN_CCCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NISO	TXP	EFBI	PXHD	Res	Res	BRSE	FD0E	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1				
0x001C	FDCAN_NBTP	NSJW[6:0]						NBRP[8:0]						NTSEG1[7:0]					Res	NTSEG2[6:0]																	
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1				
0x0020	FDCAN_TSCC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TCP[3:0]			Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TSS[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				
0x0024	FDCAN_TSCV	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TSC[15:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x0028	FDCAN_TOCC	TOP[15:0]															Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TOS[1:0]	ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x002C	FDCAN_TOCV	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TOC[15:0]																			
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					
0x0030-0x003C	Reserved	Reserved																																			
0x0040	FDCAN_ECR	Res	Res	Res	Res	Res	Res	Res	Res	CEL[7:0]							RP	REC[6:0]						TEC[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					
0x0044	FDCAN_PSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDCV[6:0]						Res	PXE	REDL	RBRRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]			LEC[2:0]						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1				
0x0048	FDCAN_TDCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TDCO[6:0]						TDCF[6:0]													
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x004C	Reserved	Reserved																																			
0x0050	FDCAN_IR	Res	Res	Res	Res	Res	Res	Res	Res	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N				
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x0054	FDCAN_IE	Res	Res	Res	Res	Res	Res	Res	Res	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE				
	Reset value									0	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 224. FDCAN 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		
0x0058	FDCAN_ILS	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PERR	BERR	MISC	TFERR	SMMSG	RXFIFO1	RXFIFO0		
	Reset value	0																									0	0	0	0	0	0	0	0	
0x005C	FDCAN_ILE	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EINT1	EINT0		
	Reset value																															0	0		
0x0060-0x007C	Reserved	Reserved																																	
0x0080	FDCAN_RXGFC	Res	Res	Res	Res	LSE[3:0]			Res	Res	Res	LSS[4:0]				Res	Res	Res	Res	Res	Res	Res	Res	Res	F0OM	F1OM	Res	Res	ANFS[1:0]	ANFE[1:0]	RRFS	RRFE			
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0		
0x0084	FDCAN_XIDAM	Res	Res	Res	EIDM[28:0]																														
	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		
0x0088	FDCAN_HPMS	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLST	Res	Res	Res	Res	FIDX[4:0]				MS[1:0]	Res	Res	Res	Res	Res	Res	BIDX[2:0]		
	Reset value																0							0	0	0	0	0				0	0	0	
0x0090	FDCAN_RXF0S	Res	Res	Res	Res	Res	Res	RF0L	F0F	Res	Res	Res	Res	Res	Res	F0P[1:0]	Res	Res	Res	Res	Res	Res	Res	Res	F0G[1:0]	Res	Res	Res	Res	Res	Res	F0FL[3:0]			
	Reset value							0	0							0	0								0	0						0	0	0	
0x0094	FDCAN_RXF0A	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	F0AI[2:0]			
	Reset value																															0	0	0	
0x0098	FDCAN_RXF1S	Res	Res	Res	Res	Res	Res	RF1L	F1F	Res	Res	Res	Res	Res	Res	F1P[1:0]	Res	Res	Res	Res	Res	Res	Res	F1G[1:0]	Res	Res	Res	Res	Res	Res	F1FL[3:0]				
	Reset value							0	0							0	0								0	0					0	0	0	0	
0x009C	FDCAN_RXF1A	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	F1AI[2:0]			
	Reset value																															0	0	0	
0x00A0-0x00BC	Reserved	Reserved																																	
0x00C0	FDCAN_TXBC	Res	Res	Res	Res	Res	Res	Res	TFQM	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																									
0x00C4	FDCAN_TXFQS	Res	Res	Res	Res	Res	Res	Res	Res	Res	TFQF	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	1	1	



Table 224. FDCAN 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
0x00C8	FDCAN_TXBRP	Res	Res	Res	Res	Res	Res	Res	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
0x00CC	FDCAN_TXBAR	Res	Res	Res	Res	Res	Res	Res	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	AR2	AR1
0x00D0	FDCAN_TXBCR	Res	Res	Res	Res	Res	Res	Res	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	CR2	CR1
0x00D4	FDCAN_TXBTO	Res	Res	Res	Res	Res	Res	Res	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	TO2	TO1
0x00D8	FDCAN_TXBCF	Res	Res	Res	Res	Res	Res	Res	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	CF2	CF1
0x00DC	FDCAN_TXBTIE	Res	Res	Res	Res	Res	Res	Res	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	TIE2	TIE1
0x00E0	FDCAN_TXBCIE	Res	Res	Res	Res	Res	Res	Res	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	CFIE2	CFIE1
0x00E4	FDCAN_TXEFS	Res	Res	Res	Res	Res	Res	TEFL	EFF	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
0x00E8	FDCAN_TXEFA	Res	Res	Res	Res	Res	Res	Res	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
0x0100	FDCAN_CKDIV	Res	Res	Res	Res	Res	Res	Res	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: Memory organization](#) for the register boundary addresses.

## 37 Universal serial bus full-speed host/device interface (USB)

### 37.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

### 37.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports both Host and Device modes
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB\_DP line)

### 37.3 USB implementation

[Table 225](#) describes the USB implementation in the devices.

**Table 225. STM32G0x1 USB implementation**

USB features <sup>(1)</sup>	USB
Host mode	X
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	2048 bytes
Dedicated packet buffer memory SRAM access scheme	32 bits
USB 2.0 Link Power Management (LPM) support in device	X
Battery Charging Detection (BCD) support for device	X
Embedded pull-up resistor on USB_DP line	X

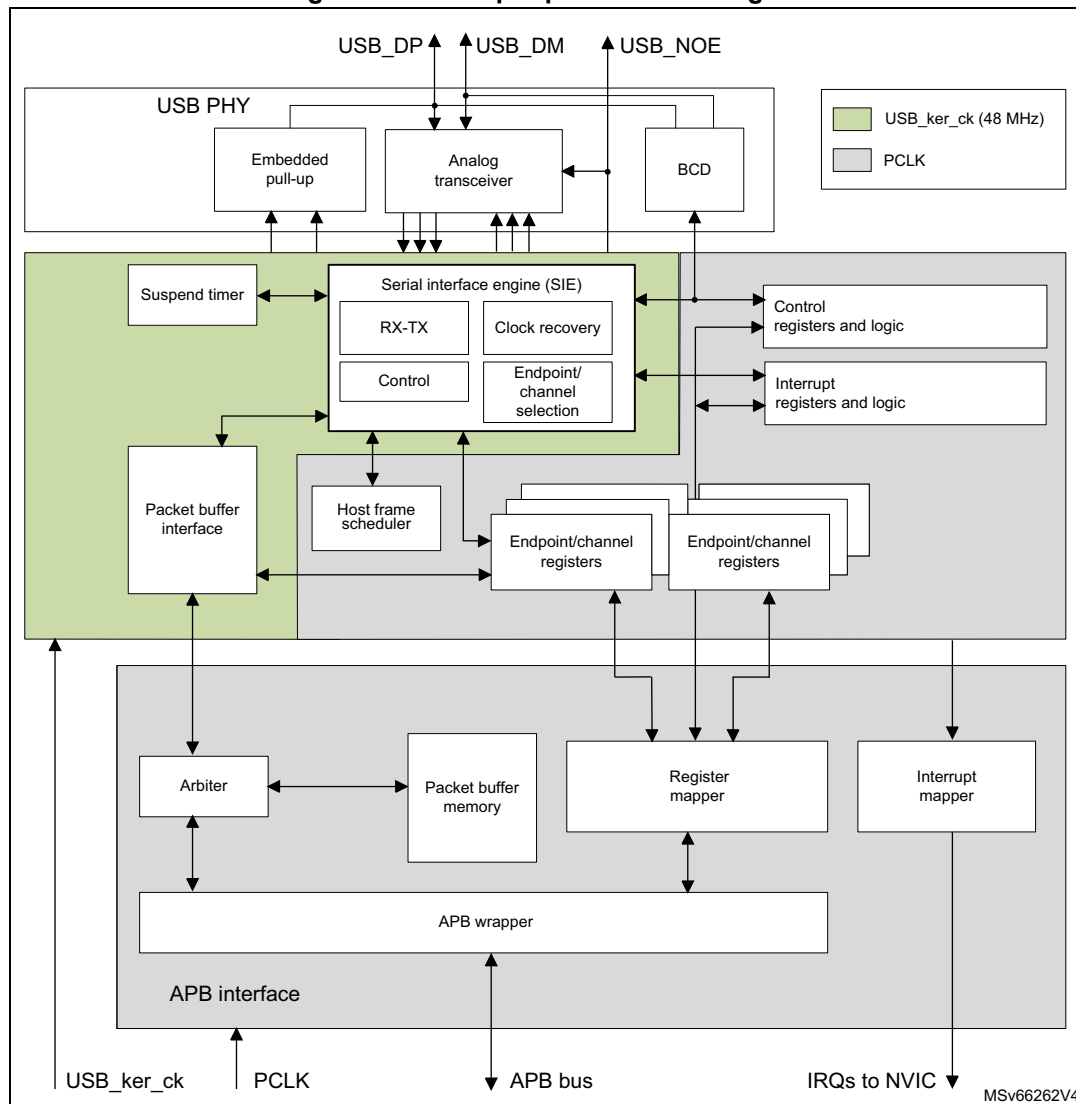
1. X= supported

## 37.4 USB functional description

### 37.4.1 USB block diagram

Figure 402 shows the block diagram of the USB peripheral.

Figure 402. USB peripheral block diagram



### 37.4.2 USB pins and internal signals

Table 226. USB input/output pins

Pin name	Pin type	Description
USB_DP	Digital input/output	D+ line

Table 226. USB input/output pins (continued)

Pin name	Pin type	Description
USB_DM	Digital input/output	D- line
USB_NOE	Digital output	NOE (output enable of data lines)

### 37.4.3 USB reset and clocks

A single reset is present on USB. The RCC allows a reset to be forced by software.

There are two clocks:

- PCLK for the APB bus interface and registers.
- USB\_ker\_ck (48 MHz) for the main protocol logic including notably the serial interface engine (SIE), see USB\_ker\_ck clock domain in block diagram.

### 37.4.4 General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB\_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

### Host mode and specific functionality

A single bit, HOST, in register USB\_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology “channel” is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

*Note: Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).*

*For LPM (link power management) this feature is not supported in Host mode.*

### 37.4.5 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB\_DP line) and support for battery charging detection (BCD), multiplexed on same USB\_DP and USB\_DM lines. The output enable control signal of the analog transceiver (active low) is provided externally on USB\_NOE. It can be used to drive some activity LED or to provide information about the actual communication direction to some other circuitry.
- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB\_Reset, data errors etc. and to endpoint

related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.

- **Timer:** this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- **Packet buffer interface:** this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- **Endpoint/channel-related registers:** each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- **Control registers:** these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- **Interrupt registers:** these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note:* \* *Endpoint/channel 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB1 bus through an APB1 interface, containing the following blocks:

- **Packet memory:** this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.
- **Arbiter:** this block accepts memory requests coming from the APB1 bus and from the USB interface. It resolves the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.
- **Register mapper:** this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB1.
- **APB1 wrapper:** this provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.
- **Interrupt mapper:** this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

### 37.4.6 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.



Host channels are divided in two categories:

- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwidth access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows

- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame, the host scheduler:

1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non-periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.

## 37.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

### 37.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

### 37.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ( $t_{\text{STARTUP}}$  specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

#### USB bus reset (RST\_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB\_DADDR register and initializing the CHEP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB\_DADDR register, and configures any other necessary endpoint.

When a RST\_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

#### USB bus reset in Host mode

In Host mode a bus reset is activated by setting the USBRST bit of the USB\_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

#### Structure and usage of packet buffers

Each bidirectional endpoint can receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request

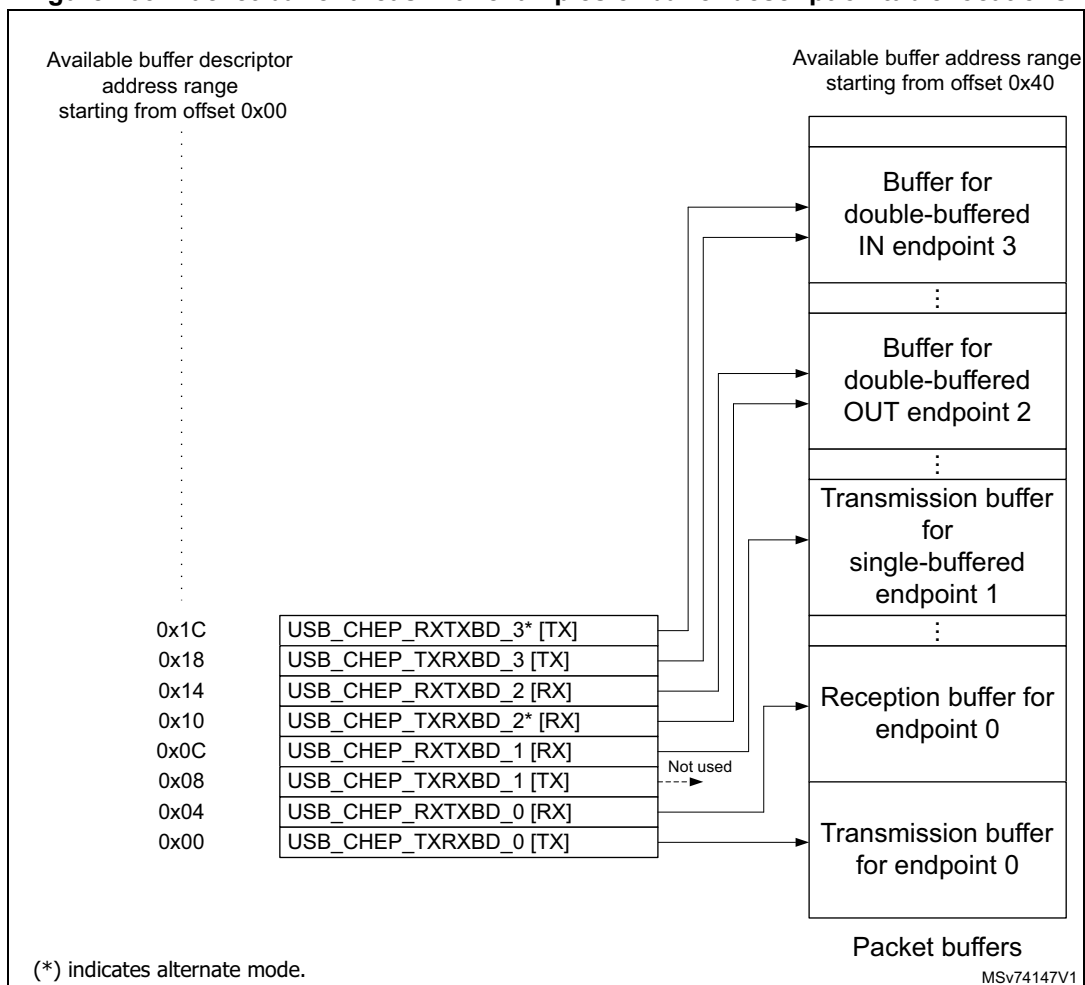
and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB1 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB1 bus. Different clock configurations are possible where the APB1 clock frequency can be higher or lower than the USB peripheral one.

*Note:* *Due to USB data rate and packet memory interface requirements, the APB1 clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.*

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-byte boundary. Buffer descriptor table entries are described in [Section 37.7: USBSRAM registers](#). If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to [Section 37.5.5: Isochronous transfers in Device mode](#) and [Section 37.5.3: Double-buffered endpoints and usage in Device mode](#) respectively). The relationship between buffer description table entries and packet buffer areas is depicted in [Figure 403](#).

For Host mode different sections explain the buffer usage model, notably [Section 37.5.6: Isochronous transfers in Host mode](#) and [Section 37.5.4: Double buffered channels: usage in Host mode](#).

**Figure 403. Packet buffer areas with examples of buffer description table locations**



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

### Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDR<sub>n</sub>\_TX/ADDR<sub>n</sub>\_RX fields in the CHEP\_TXBD<sub>n</sub> and CHEP\_RXBD<sub>n</sub> registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB\_CHEP<sub>n</sub>R register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB\_CHEP<sub>n</sub>R register and COUNT<sub>n</sub>\_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNT<sub>n</sub>\_RX must be written with the allocated buffer size using the BLSIZE and NUM\_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB\_CHEP<sub>n</sub>R and locations ADDR<sub>n</sub>\_TX/ADDR<sub>n</sub>\_RX, COUNT<sub>n</sub>\_TX/COUNT<sub>n</sub>\_RX (respectively), must not be modified by the application software, as the hardware can

change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

### Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP\_TXBD\_n (fields ADDRn\_TX and COUNTn\_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to [Structure and usage of packet buffers on page 1254](#)) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB\_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB\_CHEPnR register.

The ADDRn\_TX field in the internal register CHEP\_TXBD\_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn\_TX for COUNTn\_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB\_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB\_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn\_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

### Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn\_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn\_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the

indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB\_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR\_TX if one of the following conditions occurs:

- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

### Data reception in Device mode (OUT and SETUP packets)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn\_RX and COUNTn\_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn\_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM\_BLOCK bit fields, which are read within USB\_CHEP\_RXBD\_n content, are used to initialize BUF\_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF\_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB\_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STATRX in the USB\_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn\_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM\_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn\_RX location inside the buffer



description table entry, leaving unaffected BLSIZE and NUM\_BLOCK fields, which normally do not require to be re-written, and the USB\_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB\_ISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB\_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn\_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB\_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

### Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn\_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn\_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet an error condition is signaled via the bits VTRX and ERR\_RX if one of the following conditions occurs:

- False EOP
- Bit stuffing error
- Wrong CRC

### Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB\_CHEPnR register at each VTRX event to distinguish normal

OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS\_OUT (EPKIND in the USB\_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS\_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

### Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB\_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR\_TX set (which can generate an interrupt).



### 37.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called 'double-buffering' can be used with bulk endpoints.

When 'double-buffering' is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both 'transmission' and 'reception' packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a 'reception' double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a 'transmission' double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB\_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB\_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB\_CHEPnR register) for 'reception' double-buffered bulk endpoints or DTOGTX (bit 6 of USB\_CHEPnR register) for 'transmission' double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB\_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW\_BUF. In the following table the correspondence between USB\_CHEPnR register bits and DTOG/SW\_BUF definition is explained, for the cases of 'transmission' and 'reception' double-buffered bulk endpoints.

**Table 227. Double-buffering buffer flag definition**

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOGTX (USB_CHEPnR bit 6)	DTOGRX (USB_CHEPnR bit 14)
SW_BUF	USB_CHEPnR bit 14	USB_CHEPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW\_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

**Table 228. Bulk double-buffering memory buffers usage (Device mode)**

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (IN)	0	1	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	0	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None <sup>(1)</sup>	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None <sup>(1)</sup>	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	1	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None <sup>(1)</sup>	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None <sup>(1)</sup>	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by performing the two following actions:

- Writing UTYPE bit field at 00 in its USB\_CHEPnR register, to define the endpoint as a bulk
- Setting EPKIND bit at 1 (DBL\_BUF), in the same register.

The application software is responsible for DTOG and SW\_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL\_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL\_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB\_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB\_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL\_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW\_BUF having the same value, see [Table 228](#)). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW\_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB\_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

#### 37.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below [Table 229: Bulk double-buffering memory buffers usage \(Host mode\)](#) shows the programming settings for OUT and IN tokens.

**Table 229. Bulk double-buffering memory buffers usage (Host mode)**

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (OUT)	0	1	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	0	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None <sup>(1)</sup>	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None <sup>(1)</sup>	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (IN)	0	1	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None <sup>(1)</sup>	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None <sup>(1)</sup>	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

### 37.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB\_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering

to ease application software development, using both 'transmission' and 'reception' packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTCGRX for 'reception' isochronous endpoints, DTCGTX for 'transmission' isochronous endpoints, both in the related USB\_CHEPnR register) according to [Table 230](#).

**Table 230. Isochronous memory buffers usage**

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (IN)	0	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	USB_CHEP_RTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB\_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB\_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB\_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB\_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overflow conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB\_ISTR register anyway, in order to notify the software of the possible data corruption.

### 37.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.

The mechanism is based on a table very similar to that for Device mode. See [Table 231](#) to understand the relationship between the DTOG bit buffers and the buffer usage.

**Table 231. Isochronous memory buffers usage**

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (OUT)	0	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.	USB_CHEP_RTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations
	1	USB_CHEP_RTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.
Receive (IN)	0	USB_CHEP_RTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations	USB_CHEP_RTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB\_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID),

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

### 37.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USB\_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB\_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set SUSPRDY bit in USB\_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB\_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear SUSPEN bit of USB\_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB\_FNR register can be used according to [Table 232](#), which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST\_DCON bit in USB\_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.



**Table 232. Resume event detection**

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB\_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB\_FNR register.

*Note:* The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB\_CNTR register to 1.

### Suspend and resume in Host mode

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB\_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB\_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB\_CNTR to 1.

Below in [Table 233](#), the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.



Table 233. Resume event detection for host

[RXDP,RXDM] status	Wake-up event	Required resume software action
"00"	Not allowed (noise on bus)	Go back in Suspend mode
"10"	Full speed capable device: Not allowed (noise on bus)  Low speed device: Device remote wake-up resume	None
"01"	Full speed capable device: Device remote wake-up resume  Low speed device: Not allowed (noise on bus)	None
"11"	Not allowed (noise on bus)	Go back in Suspend mode

### 37.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common registers: interrupt and control registers. These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.
  - USB\_CNTR
  - USB\_ISTR
  - USB\_FNR
  - USB\_DADDR
  - USB\_LPMCSR
  - USB\_BCDR
- Endpoint/channel registers: endpoint/channel configuration and status
  - USB\_CHEPnR

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions.

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

#### 37.6.1 USB control register (USB\_CNTR)

Address offset: 0x40

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THR 512M
r/w															r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMA OVRM	ERRM	WKUP M	SUSP M	RST_D CONM	SOFM	ESOF M	L1REQ M	Res.	L1RE S	L2RE S	SUS PEN	SUSP RDY	PDWN	USB RST
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w	r	r/w	r/w

Bit 31 **HOST**: HOST mode

HOST bit selects between host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.

- 0: USB Device function
- 1: USB host function

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **THR512M**: 512 byte threshold interrupt mask

- 0: 512 byte threshold interrupt disabled
- 1: 512 byte threshold interrupt enabled

Bit 15 **CTRM**: Correct transfer interrupt mask

- 0: Correct transfer (CTR) interrupt disabled.
- 1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.

- Bit 14 **PMAOVRM**: Packet memory area over / underrun interrupt mask  
0: PMAOVR interrupt disabled.  
1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 13 **ERRM**: Error interrupt mask  
0: ERR interrupt disabled.  
1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 12 **WKUPM**: Wake-up interrupt mask  
0: WKUP interrupt disabled.  
1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 11 **SUSPM**: Suspend mode interrupt mask  
0: Suspend mode request (SUSP) interrupt disabled.  
1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 10 **RST\_DCONM**: USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask  
0: RESET interrupt disabled.  
1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 9 **SOFM**: Start of frame interrupt mask  
0: SOF interrupt disabled.  
1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 8 **ESOFM**: Expected start of frame interrupt mask  
0: Expected start of frame (ESOF) interrupt disabled.  
1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 7 **L1REQM**: LPM L1 state request interrupt mask  
0: LPM L1 state request (L1REQ) interrupt disabled.  
1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB\_ISTR register is set.
- Bit 6 Reserved, must be kept at reset value.
- Bit 5 **L1RES**: L1 remote wake-up / resume driver  
– Device mode  
Software sets this bit to send a LPM L1 50  $\mu$ s remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.  
0: No effect  
1: Send 50  $\mu$ s remote-wake-up signaling to host

**Bit 4 L2RES:** L2 remote wake-up / resume driver

## – Device mode

The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.

## – Host mode

Software sets this bit to send resume signaling to the device.

Software clears this bit to send end of resume to device and restart SOF generation.

In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.

0: No effect

1: Send L2 resume signaling to device

**Bit 3 SUSPEN:** Suspend state enable

## – Condition: Device mode

Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.

As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

## – Condition: Host mode

Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.

As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

**Bit 2 SUSPRDY:** Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

0: Normal operation

1: Suspend state

**Bit 1 PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.

0: Exit power down

1: Enter power down mode

**Bit 0 USBRST:** USB Reset

## – Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB. The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST\_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST\_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

0: No effect

1: USB core is under reset

## – Condition: Host mode

Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.

0: No effect

1: USB reset driven

**37.6.2 USB interrupt status register (USB\_ISTR)**

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB\_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB\_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB\_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB\_CHEPnR register (the CTR bit is actually a read only bit). For endpoint-/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB\_ISTR events by specifying the order in which software checks USB\_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LS_DCON	DCON_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THR 512
	r	r													rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA_OVR	ERR	WKUP	SUSP	RST_DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LS\_DCON**: Low speed device connected

– Host mode:

This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state.

Bit 29 **DCON\_STAT**: Device connection status

– Host mode:

This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected.

0: No device connected

1: FS or LS device connected to the host

Bits 28:17 Reserved, must be kept at reset value.

Bit 16 **THR512**: 512 byte threshold interrupt

This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel.

Bit 15 **CTR**: Completed transfer in host mode

This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR**: Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOVR interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

**Bit 13 ERR:** Error

This flag is set whenever one of the errors listed below has occurred:

**NANS:** No ANSwer. The timeout for a host response has expired.

**CRC:** Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.

**BST:** Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

**FVIO:** Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

**Bit 12 WKUP:** Wake-up

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTLR register and activates the USB\_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

**Bit 11 SUSP:** Suspend mode request

## – Device mode

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

**Bit 10 RST\_DCON:** USB reset request (Device mode) or device connect/disconnect (Host mode)

## – Device mode

This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.

## – Host mode

This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.

**Bit 9 SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine can monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB\_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

**Bit 8 ESOF:** Expected start of frame

## – Device mode

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 **L1REQ**: LPM L1 state request

– Device mode

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

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

Bit 4 **DIR**: Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.

If DIR bit = 0, VTTX bit is set in the USB\_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).

If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB\_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB\_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]**: Device Endpoint / host channel identification number

These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB\_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.

### 37.6.3 USB frame number register (USB\_FNR)

Address offset: 0x48

Reset value: 0x0000 0XXX (where X is undefined)

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

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

Bit 15 **RXDP**: Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM**: Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.



Bit 13 **LCK**: Locked

– Device mode

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]**: Lost SOF

– Device mode

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]**: Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

### 37.6.4 USB Device address (USB\_DADDR)

Address offset: 0x4C

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.	EF	ADD[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 7 **EF**: Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB\_CHEPnR registers.

Bits 6:0 **ADD[6:0]**: Device address

– Device mode

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB\_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

– Host mode

These bits contain the address transmitted with the LPM transaction

### 37.6.5 USB LPM control and status register (USB\_LPMCSR)

Address offset: 0x54

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.	BESL[3:0]				REM WAKE	Res.	LPM ACK	LPM EN
								r	r	r	r	r		rw	rw

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

Bits 7:4 **BESL[3:0]**: BESL value

– Device mode

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REM WAKE**: bRemoteWake value

– Device mode

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LPMACK**: LPM token acknowledge enable

– Device mode:

0: the valid LPM token is NYET.

1: the valid LPM token is ACK.

The NYET/ACK is returned only on a successful LPM transaction:

No errors in both the EXT token and the LPM token (else ERROR)

A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN**: LPM support enable

– Device mode

This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.

### 37.6.6 USB battery charging detector (USB\_BCDR)

Address offset: 0x58

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
DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2 DET	SDET	PDET	DC DET	SDEN	PDEN	DCD EN	BCD EN
rw								r	r	r	r	rw	rw	rw	rw

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

Bit 15 **DPPU\_DPD**: DP pull-up / DPDM pull-down

– Device mode

This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.

– Host mode

This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET**: DM pull-up detection status

– Device mode

This bit is active only during PD and gives the result of comparison between DM voltage level and  $V_{LGC}$  threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

Bit 6 **SDET**: Secondary detection (SD) status

– Device mode

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

Bit 5 **PDET**: Primary detection (PD) status

– Device mode

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 **DCDET**: Data contact detection (DCD) status

– Device mode

This bit gives the result of DCD.

0: data lines contact not detected.

1: data lines contact detected.

Bit 3 **SDEN**: Secondary detection (SD) mode enable

– Device mode

This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

- Bit 2 **PDEN**: Primary detection (PD) mode enable
  - Device mode
 

This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.
- Bit 1 **DCDEN**: Data contact detection (DCD) mode enable
  - Device mode
 

This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.
- Bit 0 **BCDEN**: Battery charging detector (BCD) enable
  - Device mode
 

This bit is set by the software to enable the BCD support within the USB Device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to 0 in order to allow the normal USB operation.

### 37.6.7 USB endpoint/channel n register (USB\_CHEPnR)

Address offset: 0x00 + 0x4 \* n, (n = 0 to 7)

Reset value: 0x0000 0000

The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an USB\_CHEPnR register is available to store the endpoint/channel specific information.

They are also reset when an USB reset is received from the USB bus or forced through bit USBRST in the CTLR register, except the VTRX and VTTX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its USB\_CHEPnR register where n is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	THREE_ERR_RX[1:0]		THREE_ERR_TX[1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	rc_w0	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VTRX	DTOG_RX	STATRX[1:0]		SETUP	UTYPE[1:0]		EP_KIND	VTTX	DTOG_TX	STATTX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

- Bit 31 Reserved, must be kept at reset value.
- Bits 30:29 **THREE\_ERR\_RX[1:0]**: Three errors for an IN transaction
- Host mode
  - This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. **THREE\_ERR\_RX** is not generated for isochronous transactions. The software can only clear this bit.
  - Coding of the received error:
    - 00: Less than 3 errors received.
    - 01: More than 3 errors received, last error is timeout error.
    - 10: More than 3 errors received, last error is data error (CRC error).
    - 11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).
- Bits 28:27 **THREE\_ERR\_TX[1:0]**: Three errors for an OUT or SETUP transaction
- Host mode
  - This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. **THREE\_ERR\_TX** is not generated for isochronous transactions. The software can only clear this bit.
  - Coding of the received error:
    - 00: Less than 3 errors received.
    - 01: More than 3 errors received, last error is timeout error.
    - 10: More than 3 errors received, last error is data error (CRC error).
    - 11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).
- Bit 26 **ERR\_RX**: Received error for an IN transaction
- Host mode
  - This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the **ERRM** bit in **USB\_CNTR** register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.
- Bit 25 **ERR\_TX**: Received error for an OUT/SETUP transaction
- Host mode
  - This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the **ERRM** bit in **USB\_CNTR** register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.
- Bit 24 **LS\_EP**: Low speed endpoint – host with HUB only
- Host mode
  - This bit is set by the software to send an LS transaction to the corresponding endpoint.
    - 0: Full speed endpoint
    - 1: Low speed endpoint
- Bit 23 **NAK**:
- Host mode
  - This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.
- Bits 22:16 **DEVADDR[6:0]**:
- Host mode
  - Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX**: USB valid transaction received

– Device mode

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB\_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only 0 can be written, writing 1 has no effect.

– Host mode

This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB\_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit

If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.

If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.

Errors can be seen via the bits ERR\_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.

**Bit 14 DTOGRX:** Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).

If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to [Section 37.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to [Section 37.5.5: Isochronous transfers in Device mode](#)). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STATRX[1:0]**: Status bits, for reception transfers

– Device mode

These bits contain information about the endpoint status, which are listed in [Table 234: Reception status encoding on page 1287](#). These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred (VTRX = 1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to [Section 37.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

These bits are the host application controls to start, retry, or abort host transactions driven by the channel.

These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

- DISABLE

DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.

- VALID

A host channel is actively trying to submit USB transaction to device only when in VALID state. VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.

VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.

- NAK

NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE

- STALL

STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.



Bit 11 **SETUP**: Setup transaction completed

## – Device mode

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.

## – Host mode

This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]**: USB type of transaction

These bits configure the behavior of this endpoint/channel as described in [Table 235: Endpoint/channel type encoding](#). Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there can be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.

The usage of isochronous channels/endpoints is explained in [Section 37.5.5: Isochronous transfers in Device mode](#)

Bit 8 **EPKIND**: endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. [Table 236](#) summarizes the different meanings.

**DBL\_BUF**: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in [Section 37.5.3: Double-buffered endpoints and usage in Device mode](#).

**STATUS\_OUT**: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered 'STALL' instead of 'ACK'. This bit can be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS\_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX**: Valid USB transaction transmitted

## – Device mode

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB\_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only 0 can be written.

## – Host mode

Same as VTRX behavior but for USB OUT and SETUP transactions.

**Bit 6 DTOGTX:** Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).

If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to [Section 37.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to [Section 37.5.5: Isochronous transfers in Device mode](#)). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

**Bits 5:4 STATTX[1:0]:** Status bits, for transmission transfers

## – Device mode

These bits contain the information about the endpoint status, listed in [Table 237](#). These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred (VTTX = 1) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to [Section 37.5.3: Double-buffered endpoints and usage in Device mode](#)).

If the endpoint is defined as isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to ‘STALL’ or ‘NAK’ for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

## – Host mode

The STATTX bits contain the information about the channel status. Refer to [Table 237](#) for the full descriptions (“Host mode” descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

**Bits 3:0 EA[3:0]:** endpoint/channel address

## – Device mode

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

## – Host mode

Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.

**Table 234. Reception status encoding**

STATRX[1:0]	Meaning
00	<b>DISABLED:</b> all reception requests addressed to this endpoint/channel are ignored.
01	<b>STALL:</b> Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.
10	<b>NAK:</b> Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request.
11	<b>VALID:</b> this endpoint/channel is enabled for reception.

**Table 235. Endpoint/channel type encoding**

UTYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

**Table 236. Endpoint/channel kind meaning**

UTYPE[1:0]		EPKIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint
11	INTERRUPT	Not used

**Table 237. Transmission status encoding**

STATTX[1:0]	Meaning
00	<b>DISABLED:</b> all transmission requests addressed to this endpoint/channel are ignored.
01	<b>STALL:</b> Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.

Table 237. Transmission status encoding (continued)

STATTX[1:0]	Meaning
10	<b>NAK:</b> Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request.
11	<b>VALID:</b> this endpoint/channel is enabled for transmission.

### 37.6.8 USB register map

The table below provides the USB register map and reset values.

**Table 238. USB register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_CHEP0R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_CHEP1R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USB_CHEP2R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_CHEP3R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USB_CHEP4R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_CHEP5R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USB_CHEP6R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	USB_CHEP7R	Res.	THREE_ERR_RX [1:0]	THREE_ERR_TX [1:0]	ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						VTRX	DTOGRX	STATRX [1:0]	SETUP	UTYPE [1:0]	EPKIND	VTTX	DTOGTX	STATTX [1:0]	EA[3:0]									
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Table 238. USB register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20-0x3F		Reserved																															
0x40	USB_CNTR	HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THR512M	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x44	USB_ISTR	Res.	LS_DCON	DCON_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THR512	CTR	PMAOVR	ERR	WKUP	SUSP	RST_DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
	Reset value	0	0	0													0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x48	USB_FNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	USB_DADDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x54	USB_LPMCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]			REMWAKE	Res.	LPMACK	LPMEN	
	Reset value																									0	0	0	0	0	0	0	0
0x58	USB_BCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSZDET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
	Reset value																									0	0	0	0	0	0	0	0

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

### 37.7 USBSRAM registers

**Note:** *The buffer descriptor table is located inside the packet buffer memory in the separate "USBSRAM" address space.*

Although the buffer descriptor table is located inside the packet buffer memory ("USBSRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USBSRAM base address. The buffer descriptor table entry associated with the USB\_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in [Structure and usage of packet buffers on page 1254](#).



### 37.7.1 Channel/endpoint transmit buffer descriptor n (USB\_CHEP\_TXRXBD\_n)

Address offset: 0x0 + 0x8 \* n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									
						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
ADDR_TX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT\_TX[9:0]**: Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB\_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR\_TX[15:0]**: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB\_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

### 37.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USB\_CHEP\_TXRXBD\_n)

Address offset: 0x0 + 0x8 \* n, (n = 0 to 7)

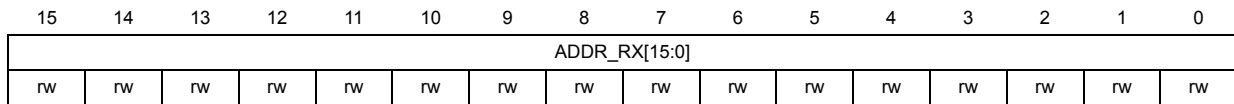
Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw





**Bit 31 BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

**Bits 30:26 NUM\_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in [Table 239](#).

**Bits 25:16 COUNT\_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB\_CHEPnR register during the last OUT/SETUP transaction addressed to it.

*Note:* Although the application only needs to read this value, it is writable.

**Bits 15:0 ADDR\_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB\_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

**Table 239. Definition of allocated buffer memory**

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
0 (00000)	Not allowed	32 bytes
1 (00001)	2 bytes	64 bytes
2 (00010)	4 bytes	96 bytes
3 (00011)	6 bytes	128 bytes
...	...	...
14 (01110)	28 bytes	480 bytes
15 (01111)	30 bytes	
16 (10000)	32 bytes	
...	...	...
29 (11101)	58 bytes	...
30 (11110)	60 bytes	992 bytes
31 (11111)	62 bytes	1023 bytes



### 37.7.3 Channel/endpoint receive buffer descriptor n (USB\_CHEP\_RTXBD\_n)

Address offset: 0x4 + 0x8 \* n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
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
ADDR_RX[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BLSIZE**: Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM\_BLOCK[4:0]**: Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in [Table 239](#).

Bits 25:16 **COUNT\_RX[9:0]**: Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB\_CHEPnR register during the last OUT/SETUP transaction addressed to it.

*Note: Although the application only needs to read this value, it is writable.*

Bits 15:0 **ADDR\_RX[15:0]**: Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB\_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

### 37.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USB\_CHEP\_RTXBD\_n)

Address offset: 0x4 + 0x8 \* n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **COUNT\_TX[9:0]**: Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB\_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR\_TX[15:0]**: Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB\_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

### 37.7.5 USBSRAM register map

The table below provides the USB register map and reset values.

**Table 240. USBSRAM register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_CHEP_TXRXBD_0	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									ADDR_TX[15:0]																
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x00	USB_CHEP_TXRXBD_0 [alternate]	BLSIZE	NUM_BLOCK[4:0]				COUNT_RX[9:0]									ADDR_RX[15:0]																	
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x04	USB_CHEP_RTXBD_0	BLSIZE	NUM_BLOCK[4:0]				COUNT_RX[9:0]									ADDR_RX[15:0]																	
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x04	USB_CHEP_RTXBD_0 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									ADDR_TX[15:0]																
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x08	USB_CHEP_TXRXBD_1	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									ADDR_TX[15:0]																
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x08	USB_CHEP_TXRXBD_1 [alternate]	BLSIZE	NUM_BLOCK[4:0]				COUNT_RX[9:0]									ADDR_RX[15:0]																	
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x0C	USB_CHEP_RTXBD_1	BLSIZE	NUM_BLOCK[4:0]				COUNT_RX[9:0]									ADDR_RX[15:0]																	
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x0C	USB_CHEP_RTXBD_1 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									ADDR_TX[15:0]																
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x38	USB_CHEP_TXRXBD_7	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									ADDR_TX[15:0]																
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x38	USB_CHEP_TXRXBD_7 [alternate]	BLSIZE	NUM_BLOCK[4:0]				COUNT_RX[9:0]									ADDR_RX[15:0]																	
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x3C	USB_CHEP_RTXBD_7	BLSIZE	NUM_BLOCK[4:0]				COUNT_RX[9:0]									ADDR_RX[15:0]																	
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
0x3C	USB_CHEP_RTXBD_7 [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]									ADDR_TX[15:0]																
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

## 38 USB Type-C<sup>®</sup>/USB Power Delivery interface (UCPD)

### 38.1 Introduction

The USB Type-C/USB Power Delivery interface complies with:

- Universal Serial Bus Type-C Cable and Connector Specification: release 2.3, Oct 2023
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.2, version 1.0, October 2023

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

### 38.2 UCPD main features

- Compliance with USB Type-C specification release 2.3
- Compliance with USB Power Delivery specifications revision 2.0 and 3.2
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - “Dead battery” Rd support
  - USB Power Delivery message transmission and reception
  - FRS (fast role swap) Rx support
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS signaling
  - FRS detection, generating an interrupt
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

### 38.3 UCPD implementation

The devices have two UCPD controllers to support two USB Type-C ports.

**Table 241. UCPD implementation<sup>(1)</sup>**

UCPD feature	UCPD1	UCPD2
Dead battery support via UCPDx_DBCC1 and UCPDx_DBCC2 external signals	X	X
UCPDx_FRSTX as alternate function pin	X	X
Fully automatic trimming	X <sup>(2)</sup>	X <sup>(2)</sup>
Discrete-component PHY support	-	-

1. "X" = supported, "-" = not supported

2. No software trimming required.

### 38.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), BIST test data (Tx and Rx), and Power Delivery Rx FRS signaling.

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD\_FRSTX pin (alternate function), external NMOS transistors that ensure low-resistance pull-down on CC lines.

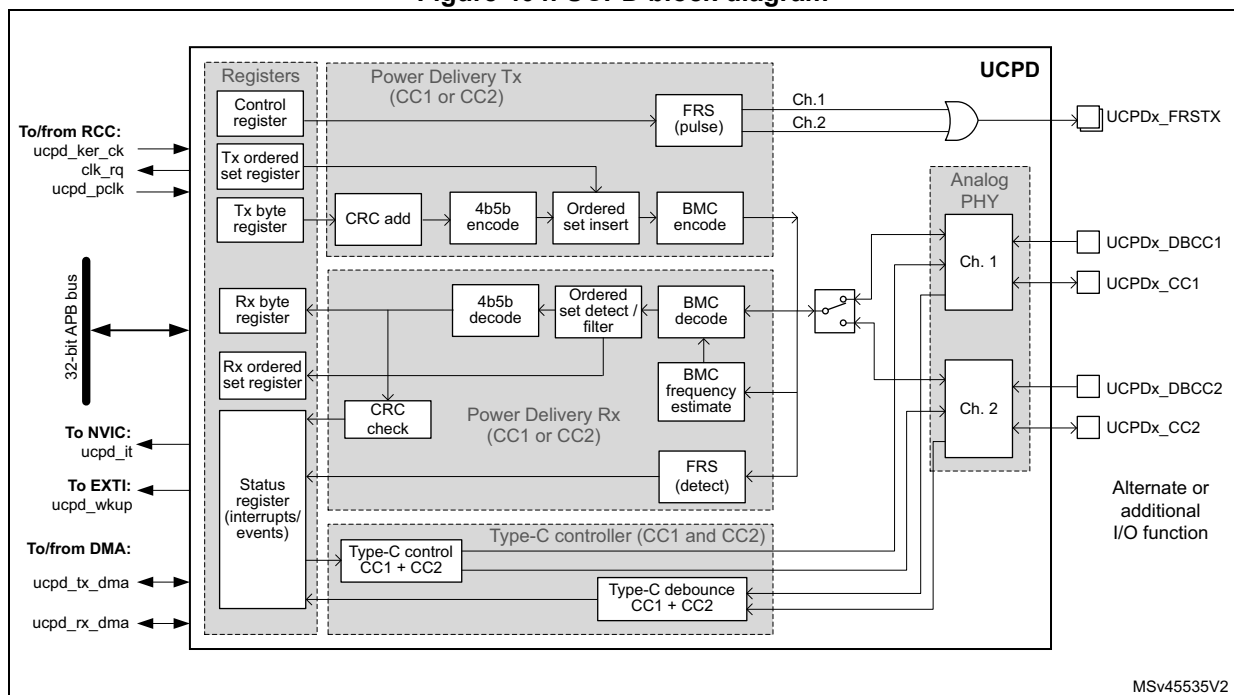
The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes "Hard Reset".

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).

In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages and FRS signaling, which allows low-power operation.

38.4.1 UCPD block diagram

Figure 404. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 242. UCPD signals on pins

Pin name	Signal type	Description
UCPDx_FRSTX	Output	USB Type-C fast role swap (FRS) signaling control, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the active CC line. A typical application has two such transistors (one per CC line) and reserves a separate I/O to drive either NMOS. Initially, the I/Os are configured as low-driving GPIOs. Upon detecting, through the Type-C state machine, the orientation of the cable attached, which determines the active CC line, the I/O of the active CC line must be set to its UCPDx_FRSTX alternate function and the I/O of the inactive CC line as low-driving GPIO.
UCPDx_CC1	Input/output	USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.
UCPDx_CC2	Input/output	USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.
UCPDx_DBCC1	Input	USB Type-C configuration control line 1 dead battery signal, to be routed to the USB Type-C connector CC1 terminal if dead battery support is required.
UCPDx_DBCC2	Input	USB Type-C configuration control line 2 dead battery signal, to be routed to the USB Type-C connector CC2 terminal if dead battery support is required.

The following table lists key internal signals.

**Table 243. UCPD internal signals**

Internal signal name	Signal type	Description
ucpd_pclk	Input	APB clock for registers
ucpd_ker_ck	Input	Kernel clock
ucpd_tx_dma	Input/Output	Rx DMA acknowledge / request
ucpd_rx_dma	Input/Output	Tx DMA acknowledge / request
ucpd_it	Output	Interrupt request (all interrupts OR-ed) connected to NVIC
ucpd_wkup	Output	Wake-up request connected to EXTI
clk_rq	Output	Clock request connected to RCC

### 38.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

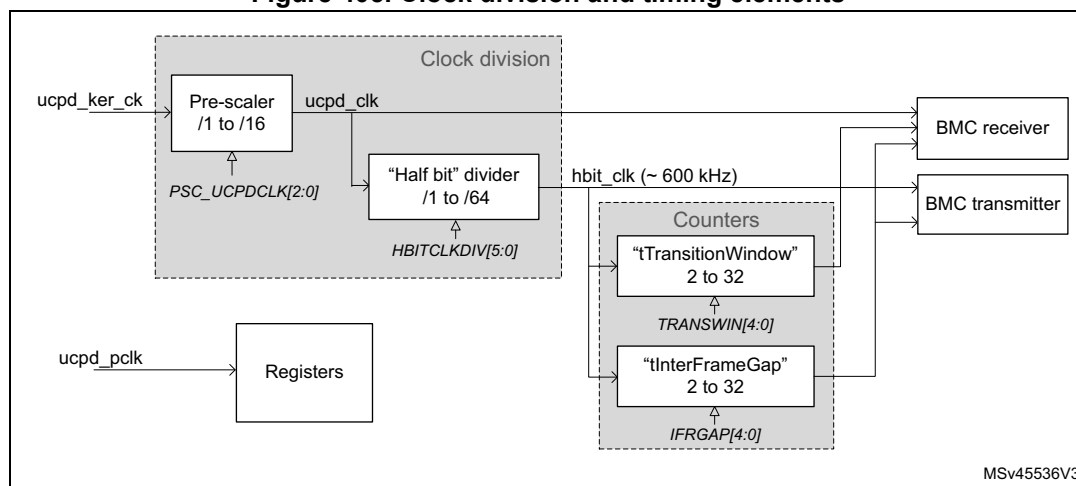
The register section is clocked with the APB clock (ucpd\_pclk).

The main functional part of the transmitter is clocked with ucpd\_clk clock, pre-scaled from the ucpd\_ker\_ck clock according to the PSC\_UCPDCLK[2:0] bitfield of the UCPD\_CFGR1 register. The main functional part of the receiver is clocked with the ucpd\_rx\_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 6 to 12 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.

**Figure 405. Clock division and timing elements**



Refer to the USB PD specification in order to set appropriate delays. For *tTransitionWindow* and especially for *tInterFrameGap*, the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.

### 38.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification.

On the transmitter side its main function is to form packets according to the defined packet format including generally:

- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:

- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

#### Symbol encoding

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.

**Table 244. 4b5b symbol encoding table**

Name	4b	5b	Symbol description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C



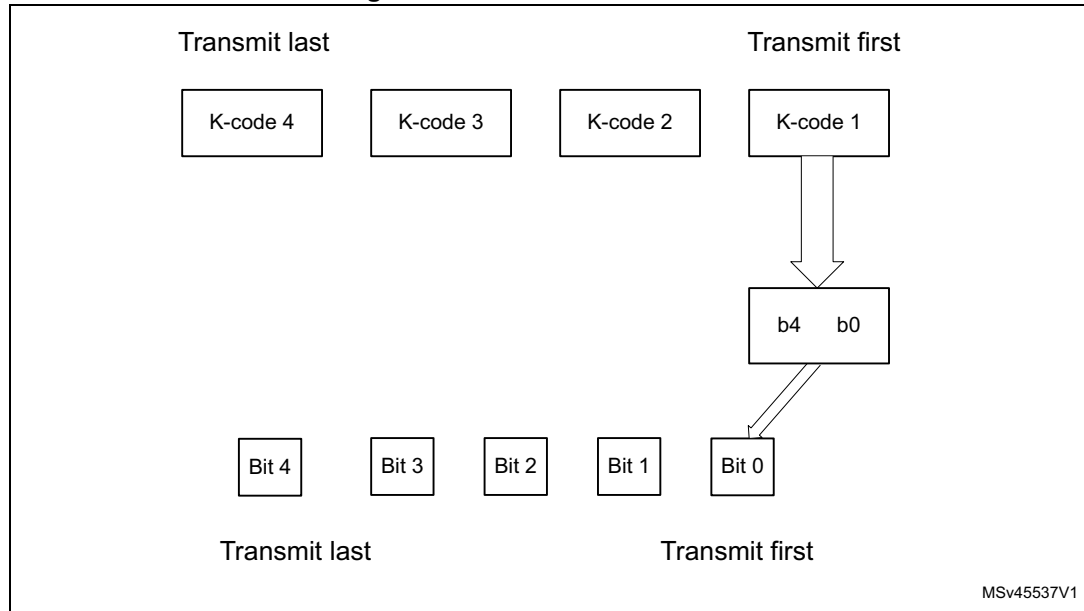
Table 244. 4b5b symbol encoding table (continued)

Name	4b	5b	Symbol description
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP
Reserved	Error	00000	Do Not Use
Reserved	Error	00001	Do Not Use
Reserved	Error	00010	Do Not Use
Reserved	Error	00011	Do Not Use
Reserved	Error	00100	Do Not Use
Reserved	Error	00101	Do Not Use
Sync-3	K-code	00110	Startsynch #3
Reserved	Error	01000	Do Not Use
Reserved	Error	01100	Do Not Use
Reserved	Error	10000	Do Not Use
Reserved	Error	11111	Do Not Use

**Ordered sets**

An ordered set consists of four K-codes as shown in the following figure.

**Figure 406. K-code transmission**



The following table lists the defined ordered sets, including all possible SOP\* sequences.

At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an ongoing Tx message.

**Table 245. Ordered sets**

Ordered set name	K-code #1	K-code #2	K-code #3	K-code #4
SOP	Sync-1	Sync-1	Sync-1	Sync-2
SOP'	Sync-1	Sync-1	Sync-3	Sync-3
SOP''	Sync-1	Sync-3	Sync-1	Sync-3
Hard Reset	RST-1	RST-1	RST-1	RST-2
Cable Reset	RST-1	Sync-1	RST-1	Sync-3
SOP*_Debug	Sync-1	RST-2	RST-2	Sync-3
SOP''_Debug	Sync-1	RST-2	Sync-3	Sync-2

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

**Table 246. Validation of ordered sets**

Status	1st code	2nd code	3rd code	4th code
Valid	Corrupt	K-code	K-code	K-code
Valid	K-code	Corrupt	K-code	K-code

**Table 246. Validation of ordered sets (continued)**

Status	1st code	2nd code	3rd code	4th code
Valid	K-code	K-code	Corrupt	K-code
Valid	K-code	K-code	K-code	Corrupt
Valid (perfect)	K-code	K-code	K-code	K-code
Not valid (example)	K-code	Corrupt	K-code	Corrupt

**Bit ordering at transmission**

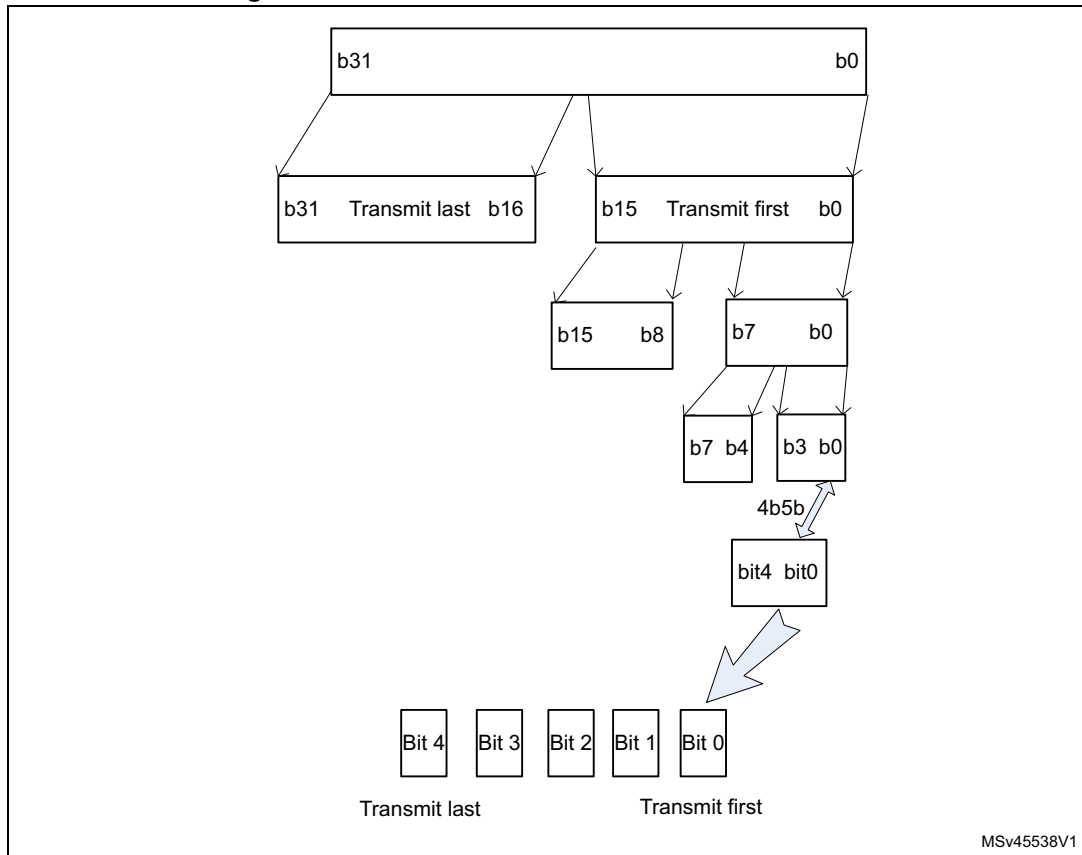
Allowed transmission data units / data sizes are in the following table.

**Table 247. Data size**

Data unit	Non-encoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20-bits
DWord	32-bits	40-bits

The bit transmission order is shown in the following figure.

**Figure 407. Transmit order for various sizes of data**



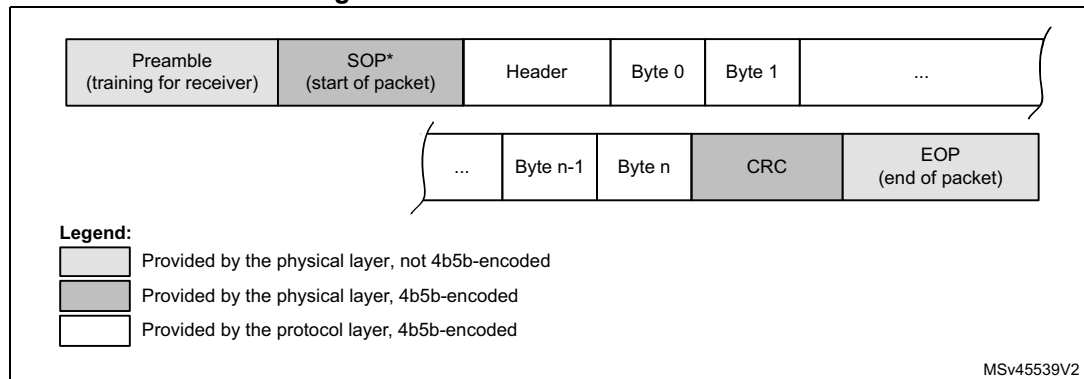
MSv45538V1

### Packet format

#### Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

Figure 408. Packet format



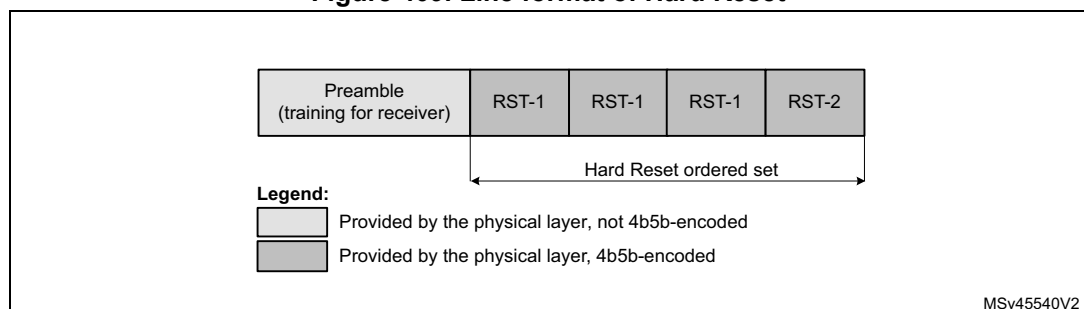
### Hard Reset

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an ongoing transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for *tInterFrameGap* time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

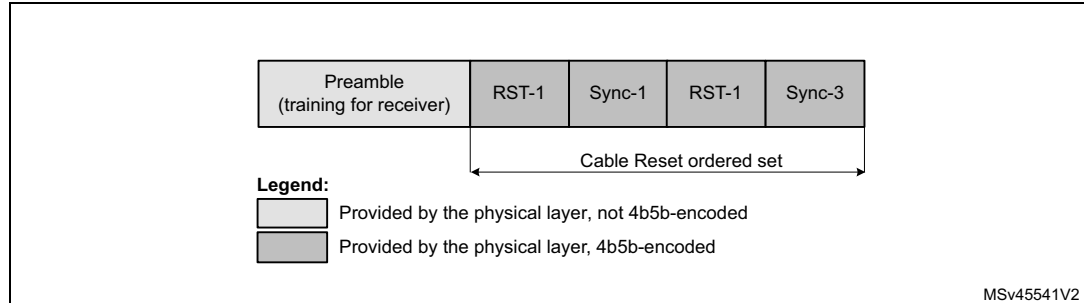
Figure 409. Line format of Hard Reset



**Cable Reset**

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

**Figure 410. Line format of Cable Reset**



**Collision avoidance**

The physical layer respects the *tInterFrameGap* delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three (*nTransitionCount*) transitions within *tTransitionWindow* (12 to 20 μs). The Power Delivery specification revision 3.1 also requires to manage the Rp value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

**Physical layer signaling schemes**

The bit are signaled with bi-phase mark coding (BMC).

**BIST**

Depending on the BIST action required by the protocol layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

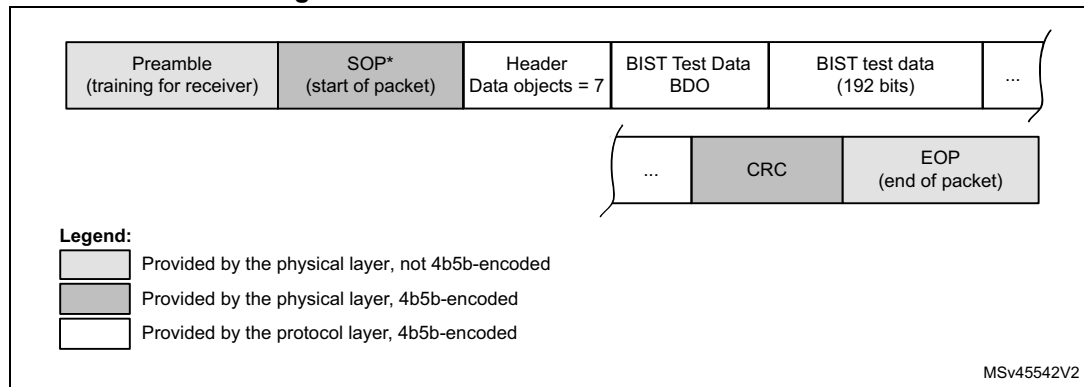
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

**BIST test data pattern**

The test data pattern is not viewed as a special case in UCPD.

The BIST test data packet frame format is shown in the following figure.

**Figure 411. BIST test data frame**



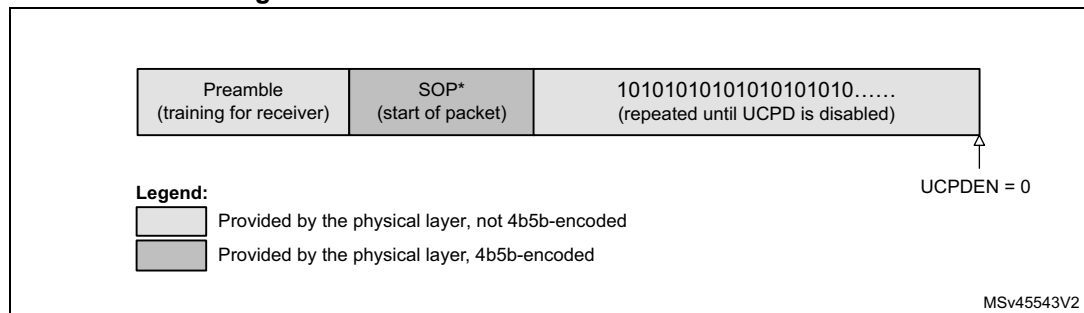
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 408: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block’s behavior) that differentiates the general packet from the BIST Test Data packet.

**BIST Carrier Mode 2**

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

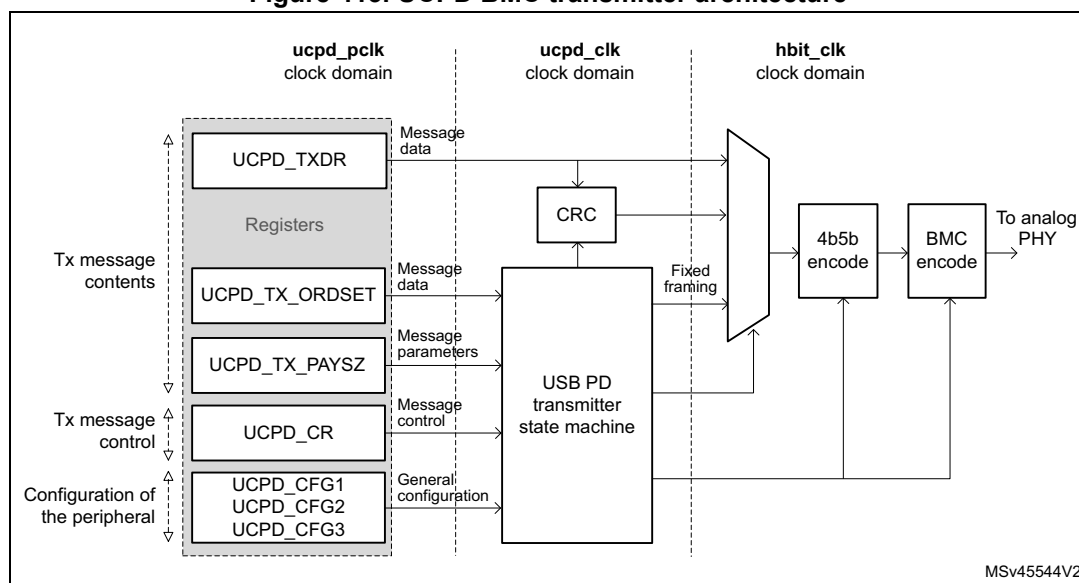
**Figure 412. BIST Carrier Mode 2 frame**



**38.4.4 UCPD BMC transmitter**

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.

Figure 413. UCPD BMC transmitter architecture



### BMC encoder

The bi-phase mark coding method is defined in the *IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 www.iec.ch* specification.

The half-bit clock `hbit_clk` is derived from `ucpd_clk` through a simple divider controlled by the `HBITCLKDIV[5:0]` bitfield of the `UCPD_CFGR1` register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

### Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

- *tInterFrameGap*: via `IFRGAP` (pre-defined value, can be altered)
- *tTransitionWindow*: via `TRANSWIN` (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

### Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of `TXMODE` field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an ongoing Tx message, which (if still in progress) is cleanly terminated by truncating the current sequence and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

### Transmitter behavior in the case of errors

The under-run condition (`TXUND` interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.

### 38.4.5 UCPD BMC receiver

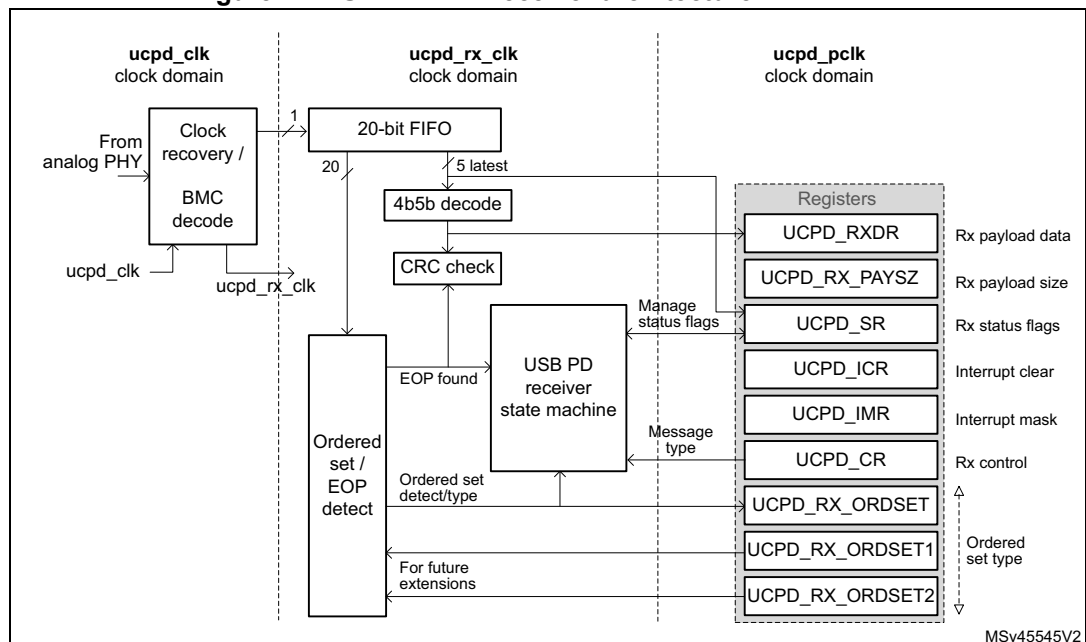
The UCPD BMC receiver performs:

- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

**Figure 414. UCPD BMC receiver architecture**



#### CRC checker

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.



**Ordered set detection**

This function detects the different ordered sets each consisting of four 5-bit K-codes.

Once we are in the preamble we opens a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP\* codes (SOP, SOP', and SOP''), but also Hard Reset, Cable Reset, SOP'\_Debug, SOP''\_Debug, and two extensions defined by registers UCPD\_RX\_ORDEXT1 and UCPD\_RX\_ORDEXT2.

**EOP detection and Hard Reset exception handling**

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection can be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an "early EOP" is detected, the truncated Rx message is immediately discarded.

**Truncated or corrupted message exception**

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

- the line becomes static for a time significantly longer than one "UI" period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
- the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

**Short preamble or incomplete ordered set exception**

In the exceptional case of the receiver seeing less than half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

**38.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)**

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it still presents the "dead battery" Rd, provided that UCPDx\_DBCC1 and UCPDx\_DBCC2 pins are each connected to UCPDx\_CC1 and UCPDx\_CC2 pins, respectively.

If dead battery behavior is not required (for example for source only products), then UCPDx\_DBCC1 and UCPDx\_DBCC2 pins must both be tied to ground.

After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0] before setting the UCPDx\_STROBE bits of the SYSCFG1\_CFGR1 register to activate this behavior.

Use of Standby low-power mode is possible for sinks in the unattached state.

### 38.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC\_VSTATE\_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

### 38.4.8 UCPD fast role swap (FRS)

#### FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the I/O configured as UCPD\_FRSTX (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

#### FRS detection

FRS monitoring is enabled by setting the bit FRSRXEN, after writing PHYCCSEL that selects the active CC line depending on the cable orientation detected.

### 38.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle UCPD1\_TXDR and UCPD1\_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

### 38.4.10 Wake-up from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.

The events causing the wake-up can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

### 38.5 UCPD programming sequences

The normal sequence of use of the UCPD unit is:

1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
  - On demand from protocol layer, send Tx message
  - Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

#### 38.5.1 Initialization phase

Use the following sequence for a clean startup:

1. Prepare all initial clock divider values, by writing the UCPD\_CFG register.
2. Enable the unit, by setting the UCPDEN bit.

#### 38.5.2 Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

**Table 248. Coding for ANAMODE, ANASUBMODE and link with TYPEC\_VSTATE\_CCx**

ANAMODE	ANASUBMODE[1:0]	Notes	TYPEC_VSTATE_CCx[1:0]			
			00	01	10	11
0: Source	00: Disabled	Disabled	N/A			
	01: Default USB Rp	-	vRa[Def]	vRd[Def]	vOPEN[Def]	N/A
	10: 1.5A Rp	-	vRa[1.5]	vRd[1.5]	vOPEN[1.5]	
	11: 3.0A Rp	-	vRa[3.0]	vRd[3.0]	vOPEN[3.0]	
1: Sink	xx	-	vRa	vRd-USB	vRd-1.5	vRd-3.0

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

*Note: The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and, when in source mode, determines VCONN generation and VBUS state (ON/OFF/+voltage level); discharge). UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be*



*connected to the CC line), but the application needs these inputs and controls, to function correctly.*

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC\_VSTATE fields represent. Note that before programming, the PHY is effectively off.
2. Read TYPEC\_VSTATE\_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).
3. In the case of no connection, wait for a connection event.
4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.
5. When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHYEVT1/2 occurs and the TYPEC\_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):

**Table 249. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)**

Type-C state	ANAMODE; ANASUBMODE[1:0]	CCENABLE	PHYCSEL	RDCH	CC[x] VCONNEN <sup>(1)</sup>	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)		00: [neither]	PHYEVT 1: [VRd- 3A0]	Wait for sink attach detect ; seen on CC1 [EVT1]
Attachwait. SRC						PHYEVT 2: [VRa]	Attachwait started (100- 200 ms) ; now also see the Ra => requesting VCONN
Attached. SRC [VCONN => CC2]	0:Source; 11:Rp3A0 [SinkTxOK]	01: CC2 disable (possible and recommend ed due to external VCONN switch)	0 [Rd on CC1]	0: [Normal]	10: [CC2 active]	Timer (100 ms) and no PHYEVT x	Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEVT1
						SW timers (SinkTxN G)	Source wants to initiate message sequence (SinkTxNG condition set first)
							Source finished message sequence (SinkTxOK condition afterwards)
Unattached wait. SRC	0:Source; 11:Rp3A0 [SinkTxOK]	11:both enabled	0 (do not care)	1: [discha rge]	00: [neither]	PHYEVT 1: [VOpen- 3A0]	Wait for Sink disconnected (Vopen on CC1)
Unattached. SRC	0:Source; 11:Rp3A0					> 0.8V detection	Discharge VCONN [CC2] actively [Rdch]; to < 0.8V
Unattached. SRC	0:Source; 11:Rp3A0			0: [Normal]			[Details as first line of table]

1. Two GPIOs to enable VCONN through external load switch components

### 38.5.3 USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD\_TX\_ORDSET and UCPD\_TX\_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD\_TXDR register.

This re-iterates until the entire payload of data is transmitted.

Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

### Hard Reset transmission

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD\_CR register forces the internal state machine to generate the correct sequence. The value of UCPD\_TX\_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

### Use of DMA for transmission

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD\_CR register.

For each message:

- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

## 38.5.4 USB PD receive

Notification of start of the receive message sequence is triggered by an interrupt on UCPD\_SR (bit RXORDDDET).

The information is recovered by reading:

- UCPD\_RX\_SOP (on interrupt RXORDDDET)
- UCPD\_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD\_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD\_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:

- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set

### Use of DMA for reception

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD\_CR register.

Whenever a Rx message is expected:

- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

## 38.6 UCPD low-power modes

A summary of low-power modes is shown below in [Table 250: Effect of low power modes on the UCPD](#).

**Table 250. Effect of low power modes on the UCPD**

Mode	Description
Sleep	No effect
Stop	Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.
Standby	UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.
Unpowered	Dead battery pull-downs remain active.

The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC\_VSTATE\_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wake-up from Stop mode is enabled by setting the WUPEN bit in the UCPD\_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which can later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which can potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which can potentially generate an FRS signaling detection event (that is, FRSEVT) later

In order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

### 38.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

**Table 251. UCPD interrupt requests**

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
FRS detection	FRSEVT	Set FRSEVTCF	FRSEVTIE
Type C voltage level change on CC2	TYPECEVT2	Set TYPECEVT2CF	TYPECEVT2IE
Type C voltage level change on CC1	TYPECEVT1	Set TYPECEVT1CF	TYPECEVT1IE
Rx message received	RXMSGEND	Set RXMSGENDCF	RXMSGENDIE
Rx data overflow	RXOVR	Set RXOVRCF	RXOVR
Rx Hard Reset detected	RXHRSTDET	Set RXHRSTDETCF	RXHRSTDETIE
Rx ordered set (4 K-codes) detected	RXORDDET	Set RXORDDETCF	RXORDDETIE
Receive data register not empty	RXNE	Read data in UCPD_RXDR	RXNEIE
Tx data underrun	TXUND	Set TXUNDCF	TXUNDIE
Hard Reset sent	HRSTSENT	Set HRSTSENTCF	HRSTSENTIE
Hard Reset discarded	HRSTDISC	Set HRSTDISCCF	HRSTDISCIE
Transmit message aborted	TXMSGABT	Set TXMSGABTCF	TXMSGABTIE
Transmit message sent	TXMSGSENT	Set TXMSGSENTCF	TXMSGSENTIE
Transmit message discarded	TXMSGDISC	Set TXMSGDISCCF	TXMSGDISCIE
Transmit data required	TXIS	Write data to the UCPD_TXDR register	TXISIE

When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD\_SR register.

Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD\_ICR register.



## 38.8 UCPD registers

### 38.8.1 UCPD configuration register 1 (UCPD\_CFGR1)

Address offset: 0x000

Reset value: 0x0000 0000

General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]								PSC_UCPDCLK[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
TRANSWIN[4:0]				IFRGAP[4:0]				HBITCLKDIV[5:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UCPDEN**: UCPD peripheral enable

General enable of the UCPD peripheral.

0: Disable

1: Enable

Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

Bit 30 **RXDMAEN**: Reception DMA mode enable

When set, the bit enables DMA mode for reception.

0: Disable

1: Enable

Bit 29 **TXDMAEN**: Transmission DMA mode enable

When set, the bit enables DMA mode for transmission.

0: Disable

1: Enable

Bits 28:20 **RXORDSETEN[8:0]**: Receiver ordered set enable

The bitfield determines the types of ordered sets that the receiver must detect. When set/cleared, each bit enables/disables a specific function:

0bXXXXXXXXX1: SOP detect enabled

0bXXXXXXXX1X: SOP' detect enabled

0bXXXXXX1XX: SOP'' detect enabled

0bXXXXX1XXX: Hard Reset detect enabled

0bXXXX1XXXX: Cable Detect reset enabled

0bXXX1XXXXX: SOP'\_Debug enabled

0bXX1XXXXXX: SOP''\_Debug enabled

0bX1XXXXXXX: SOP extension#1 enabled

0b1XXXXXXX: SOP extension#2 enabled

- Bits 19:17 **PSC\_UCPDCLK[2:0]**: Pre-scaler division ratio for generating `ucpd_clk`  
The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD peripheral clock (`ucpd_clk`).  
0x0: 1 (bypass)  
0x1: 2  
0x2: 4  
0x3: 8  
0x4: 16  
It is recommended to use the pre-scaler so as to set the `ucpd_clk` frequency in the range from 6 to 9 MHz.
- Bit 16 Reserved, must be kept at reset value.
- Bits 15:11 **TRANSWIN[4:0]**: Transition window duration  
The bitfield determines the division ratio (the bitfield value minus one) of a `hbit_clk` divider producing *tTransitionWindow* interval.  
0x00: Not supported  
0x01: 2  
0x09: 10 (recommended)  
0x1F: 32  
Set a value that produces an interval of 12 to 20 us, taking into account the `ucpd_clk` frequency and the `HBITCLKDIV[5:0]` bitfield setting.
- Bits 10:6 **IFRGAP[4:0]**: Division ratio for producing inter-frame gap timer clock  
The bitfield determines the division ratio (the bitfield value minus one) of a `ucpd_clk` divider producing inter-frame gap timer clock (*tInterFrameGap*).  
0x00: Not supported  
0x01: 2  
0x0D: 14  
0x0E: 15  
0x0F: 16  
0x1F: 32  
The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value. The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios above 15 for Tx clock above nominal.
- Bits 5:0 **HBITCLKDIV[5:0]**: Division ratio for producing half-bit clock  
The bitfield determines the division ratio (the bitfield value plus one) of a `ucpd_clk` divider producing half-bit clock (`hbit_clk`).  
0x00: 1 (bypass)  
0x1A: 27  
0x3F: 64

### 38.8.2 UCPD configuration register 2 (UCPD\_CFGR2)

Address offset: 0x004

Reset value: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
												rw	rw	rw	rw

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

Bit 3 **WUPEN**: Wake-up from Stop mode enable

Setting the bit enables the UCPD\_ASYNC\_INT signal.

0: Disable

1: Enable

Bit 2 **FORCECLK**: Force ClkReq clock request

0: Do not force clock request

1: Force clock request

Bit 1 **RXFILT2N3**: BMC decoder Rx pre-filter sampling method

Number of consistent consecutive samples before confirming a new value.

0: 3 samples

1: 2 samples

Bit 0 **RXFILTDIS**: BMC decoder Rx pre-filter enable

0: Enable

1: Disable

The sampling clock is that of the receiver (that is, after pre-scaler).

### 38.8.3 UCPD control register (UCPD\_CR)

Address offset: 0x00C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN
										rw	rw		rw	rs	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBATTEN	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
rw				rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw	rw

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

Bit 21 **CC2TCDIS**: CC2 Type-C detector disable

The bit disables the Type-C detector on the CC2 line.

- 0: Enable
- 1: Disable

When enabled, the Type-C detector for CC2 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 20 **CC1TCDIS**: CC1 Type-C detector disable

The bit disables the Type-C detector on the CC1 line.

- 0: Enable
- 1: Disable

When enabled, the Type-C detector for CC1 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RDCH**: Rdch condition drive

The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus associated with VCONN), by remaining set during the source-only *UnattachedWait.SRC* state, to respect the Type-C state. Refer to "USB Type-C ECN for Source VCONN Discharge". The CCENABLE[1:0] bitfield must be set accordingly, too.

- 0: No effect
- 1: Rdch condition drive

Changing the bit value only takes effect upon setting the UCPDx\_STROBE bit of the SYSCFG\_CFGR1 register.

Bit 17 **FRSTX**: FRS Tx signaling enable.

Setting the bit enables FRS Tx signaling.

- 0: No effect
- 1: Enable

The bit is cleared by hardware after a delay respecting the USB Power Delivery specification Revision 3.1.

- Bit 16 **FRSRXEN**: FRS event detection enable  
Setting the bit enables FRS Rx event (FRSEVT) detection on the CC line selected through the PHYCCSEL bit. 0: Disable  
1: Enable  
Clear the bit when the device is attached to an FRS-incapable source/sink.
- Bit 15 **DBATTEN**: Dead battery function enable  
The bit takes effect upon setting the USBPDstrobe bit of the SYS\_CONFIG register.  
0: Disable  
1: Enable  
Dead battery function only operates if the external circuit is appropriately configured.
- Bit 14 Reserved, must be kept at reset value.
- Bit 13 Reserved, must be kept at reset value.
- Bit 12 Reserved, must be kept at reset value.
- Bits 11:10 **CCENABLE[1:0]**: CC line enable  
This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to ANAMODE and ANASUBMODE[1:0] setting.  
0x0: Disable both PHYs  
0x1: Enable CC1 PHY  
0x2: Enable CC2 PHY  
0x3: Enable CC1 and CC2 PHY  
A single line PHY can be enabled when, for example, the other line is driven by VCONN via an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.
- Bit 9 **ANAMODE**: Analog PHY operating mode  
The bit takes effect upon setting the UCPDx\_STROBE bit of the SYS\_CONFIG register.  
0: Source  
1: Sink  
The use of CC1 and CC2 depends on CCENABLE. Refer to [Table 248: Coding for ANAMODE, ANASUBMODE and link with TYPEC\\_VSTATE\\_CCx](#) for the effect of this bitfield in conjunction with ANASUBMODE[1:0].
- Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode  
Refer to [Table 248: Coding for ANAMODE, ANASUBMODE and link with TYPEC\\_VSTATE\\_CCx](#) for the effect of this bitfield.
- Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling  
0: Use CC1 IO for Power Delivery communication  
1: Use CC2 IO for Power Delivery communication  
The selection depends on the cable orientation as discovered at attach.
- Bit 5 **PHYRXEN**: USB Power Delivery receiver enable  
0: Disable  
1: Enable  
Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode

Determines the mode of the receiver.

0: Normal receive mode

1: BIST receive mode (BIST test data mode)

When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset

0: No effect

1: Start Tx Hard Reset message

The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet

0: No effect

1: Start Tx packet transmission

The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet

Writing the bitfield triggers the action as follows, depending on the value:

0x0: Transmission of Tx packet previously defined in other registers

0x1: Cable Reset sequence

0x2: BIST test sequence (BIST Carrier Mode 2)

Others: invalid

From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).

### 38.8.4 UCPD interrupt mask register (UCPD\_IMR)

Address offset: 0x010

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	FRSEVTIE	Res.	Res.	Res.	Res.
											1				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMGDISCIE	TXISIE
r/w	r/w		r/w	r/w	r/w	r/w	r/w		r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTIE**: FRSEVT interrupt enable

0: Disable

1: Enable

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable

0: Disable

1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable

0: Disable

1: Enable

Bit 11 **RXOVRIE**: RXOVR interrupt enable

0: Disable

1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable

0: Disable

1: Enable

Bit 9 **RXORDDIE**: RXORDDIE interrupt enable

0: Disable

1: Enable

Bit 8 **RXNEIE**: RXNE interrupt enable

0: Disable

1: Enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUNDIE**: TXUND interrupt enable

0: Disable

1: Enable

Bit 5 **HRSTSENTIE**: HRSTSENT interrupt enable

0: Disable

1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable

0: Disable

1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable

0: Disable

1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable

0: Disable

1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable

0: Disable

1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable  
 0: Disable  
 1: Enable

### 38.8.5 UCPD status register (UCPD\_SR)

Address offset: 0x014

Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD\_IMR 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.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]	
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
r	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVT**: FRS detection event  
 The flag is cleared by setting the FRSEVTCF bit.  
 0: No new event  
 1: New FRS receive event occurred

Bits 19:18 **TYPEC\_VSTATE\_CC2[1:0]**: CC2 line voltage level  
 The status bitfield indicates the voltage level on the CC2 line in its steady state.  
 0x0: Lowest  
 0x1: Low  
 0x2: High  
 0x3: Highest  
 The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16 **TYPEC\_VSTATE\_CC1[1:0]**:  
 The status bitfield indicates the voltage level on the CC1 line in its steady state.  
 0x0: Lowest  
 0x1: Low  
 0x2: High  
 0x3: Highest  
 The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.



- Bit 15 **TYPECEVT2**: Type-C voltage level event on CC2 line  
The flag indicates a change of the TYPEC\_VSTATE\_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.  
0: No new event  
1: A new Type-C event
- Bit 14 **TYPECEVT1**: Type-C voltage level event on CC1 line  
The flag indicates a change of the TYPEC\_VSTATE\_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.  
0: No new event  
1: A new Type-C event
- Bit 13 **RXERR**: Receive message error  
The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.  
0: No error detected  
1: Error(s) detected
- Bit 12 **RXMSGEND**: Rx message received  
The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.  
0: No new Rx message received  
1: A new Rx message received  
The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.
- Bit 11 **RXOVR**: Rx data overflow detection  
The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVRDCF bit.  
0: No overflow  
1: Overflow  
The buffer overflow can occur if the received data are not read fast enough.
- Bit 10 **RXHRSTDET**: Rx Hard Reset receipt detection  
The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDETCF bit.  
0: Hard Reset not received  
1: Hard Reset received
- Bit 9 **RXORDDDET**: Rx ordered set (4 K-codes) detection  
The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD\_RX\_ORDSET register. It is cleared by setting the RXORDDDETCF bit.  
0: No ordered set detected  
1: A new ordered set detected
- Bit 8 **RXNE**: Receive data register not empty detection  
The flag indicates that the UCPD\_RXDR register is not empty. It is automatically cleared upon reading UCPD\_RXDR.  
0: Rx data register empty  
1: Rx data register not empty
- Bit 7 Reserved, must be kept at reset value.

- Bit 6 **TXUND**: Tx data underrun detection  
The flag indicates that the Tx data register (UCPD\_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.  
0: No Tx data underrun detected  
1: Tx data underrun detected
- Bit 5 **HRSTSENT**: Hard Reset message sent  
The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENTCF bit.  
0: No Hard Reset message sent  
1: Hard Reset message sent
- Bit 4 **HRSTDISC**: Hard Reset discarded  
The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.  
0: No Hard Reset discarded  
1: Hard Reset discarded
- Bit 3 **TXMSGABT**: Transmit message abort  
The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.  
0: No transmit message abort  
1: Transmit message abort
- Bit 2 **TXMSGSENT**: Message transmission completed  
The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSENTCF bit.  
0: No Tx message completed  
1: Tx message completed  
In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.
- Bit 1 **TXMSGDISC**: Message transmission discarded  
The flag indicates that a message transmission was dropped. The flag is cleared by setting the TXMSGDISCCF bit.  
0: No Tx message discarded  
1: Tx message discarded  
Transmission of a message can be dropped if there is a concurrent receive in progress or at excessive noise on the line. After a Tx message is discarded, the flag is only raised when the CC line becomes idle.
- Bit 0 **TXIS**: Transmit interrupt status  
The flag indicates that the UCPD\_TXDR register is empty and new data write is required (as the amount of data sent has not reached the payload size defined in the TXPAYSZ bitfield). The flag is cleared with the data write into the UCPD\_TXDR register.  
0: New Tx data write not required  
1: New Tx data write required

### 38.8.6 UCPD interrupt clear register (UCPD\_ICR)

Address offset: 0x018

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 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.	FRSEVTCF	Res.	Res.	Res.	Res.
											w				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
w	w		w	w	w	w			w	w	w	w	w	w	

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTCF**: FRS event flag (FRSEVT) clear

Setting the bit clears the FRSEVT flag in the UCPD\_SR register.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2CF**: Type-C CC2 line event flag (TYPECEVT2) clear

Setting the bit clears the TYPECEVT2 flag in the UCPD\_SR register

Bit 14 **TYPECEVT1CF**: Type-C CC1 event flag (TYPECEVT1) clear

Setting the bit clears the TYPECEVT1 flag in the UCPD\_SR register

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear

Setting the bit clears the RXMSGEND flag in the UCPD\_SR register.

Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear

Setting the bit clears the RXOVR flag in the UCPD\_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detect flag (RXHRSTDET) clear

Setting the bit clears the RXHRSTDET flag in the UCPD\_SR register.

Bit 9 **RXORDETCF**: Rx ordered set detect flag (RXORDET) clear

Setting the bit clears the RXORDET flag in the UCPD\_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear

Setting the bit clears the TXUND flag in the UCPD\_SR register.

Bit 5 **HRSTSENTCF**: Hard reset send flag (HRSTSENT) clear

Setting the bit clears the HRSTSENT flag in the UCPD\_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear

Setting the bit clears the HRSTDISC flag in the UCPD\_SR register.

- Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear  
Setting the bit clears the TXMSGABT flag in the UCPD\_SR register.
- Bit 2 **TXMSGSENTCF**: Tx message send flag (TXMSGSENT) clear  
Setting the bit clears the TXMSGSENT flag in the UCPD\_SR register.
- Bit 1 **TXMSGDISCCF**: Tx message discard flag (TXMSGDISC) clear  
Setting the bit clears the TXMSGDISC flag in the UCPD\_SR register.
- Bit 0 Reserved, must be kept at reset value.

### 38.8.7 UCPD Tx ordered set type register (UCPD\_TX\_ORDSETR)

Address offset: 0x01C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

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.	TXORDSET[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXORDSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **TXORDSET[19:0]**: Ordered set to transmit

The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.

### 38.8.8 UCPD Tx payload size register (UCPD\_TX\_PAYSZR)

Address offset: 0x020

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

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

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

Bits 9:0 **TXPAYSZ[9:0]**: Payload size yet to transmit

The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD\_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission.

0x2: 2 bytes - the size of Control message from the protocol layer

0x6: 6 bytes - the shortest Data message allowed from the protocol layer

0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer

0x106: 262 bytes - the longest possible extended message

0x3FF: 1024 bytes - the longest possible payload (for future expansion)

### 38.8.9 UCPD Tx data register (UCPD\_TXDR)

Address offset: 0x024

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

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

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

Bits 7:0 **TXDATA[7:0]**: Data byte to transmit

### 38.8.10 UCPD Rx ordered set register (UCPD\_RX\_ORDSETR)

Address offset: 0x028

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.	RXSOPKINVALID[2:0]		RXSOP30F4		RXORDSET[2:0]		
									r	r	r	r	r	r	r

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



Bits 6:4 **RXSOPKINVALID[2:0]:**

The bitfield is for debug purposes only.  
 0x0: No K-code corrupted  
 0x1: First K-code corrupted  
 0x2: Second K-code corrupted  
 0x3: Third K-code corrupted  
 0x4: Fourth K-code corrupted  
 Others: Invalid

Bit 3 **RXSOP3OF4:**

The bit indicates the number of correct K-codes. For debug purposes only.  
 0: 4 correct K-codes out of 4  
 1: 3 correct K-codes out of 4

Bits 2:0 **RXORSET[2:0]:** Rx ordered set code detected

0x0: SOP code detected in receiver  
 0x1: SOP' code detected in receiver  
 0x2: SOP" code detected in receiver  
 0x3: SOP'\_Debug detected in receiver  
 0x4: SOP"'\_Debug detected in receiver  
 0x5: Cable\_Reset detected in receiver  
 0x6: SOP extension#1 detected in receiver  
 0x7: SOP extension#2 detected in receiver

### 38.8.11 UCPD Rx payload size register (UCPD\_RX\_PAYSZR)

Address offset: 0x02C

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.	RXPAYSZ[9:0]										
						r	r	r	r	r	r	r	r	r	r	

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

Bits 9:0 **RXPAYSZ[9:0]:** Rx payload size received

This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD\_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).  
 0x2: 2 bytes - the size of Control message from the protocol layer  
 0x6: 6 bytes - the shortest Data message allowed from the protocol layer  
 0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer  
 0x106: 262 bytes - the longest possible extended message  
 0x3FF: 1024 bytes - the longest possible payload (for future expansion)  
 The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

**38.8.12 UCPD receive data register (UCPD\_RXDR)**

Address offset: 0x030

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.	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]**: Data byte received

**38.8.13 UCPD Rx ordered set extension register 1 (UCPD\_RX\_ORDEXTR1)**

Address offset: 0x034

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **RXSOPX1[19:0]**: Ordered set 1 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

### 38.8.14 UCPD Rx ordered set extension register 2 (UCPD\_RX\_ORDEXTR2)

Address offset: 0x038

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	RXSOPX2[19:16]				17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.					r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RXSOPX2[15:0]																	
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w		

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

Bits 19:0 **RXSOPX2[19:0]**: Ordered set 2 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

### 38.8.15 UCPD register map

Table 252. UCPD register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	UCPD_CFGR1	UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]								PSC_UCPDCLK[2:0]		Res.	TRANSWIN[4:0]				IFRGAP[4:0]				HBITCLKDIV[5:0]									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	UCPD_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																													0	0	0	0
0x008	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.	Res.
0x00C	UCPD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0





Table 252. UCPD 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
0x010	UCPD_IMR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FRSEVTIE	Res	Res	Res	Res	TYPECEVT2IE	TYPECEVT1IE	Res	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
	Reset value												0					0	0		0	0	0	0	0		0	0	0	0	0	0	
0x014	UCPD_SR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FRSEVT	TYPEC_VSTATE_CC2[1:0]	TYPEC_VSTATE_CC1[1:0]	Res	Res	TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x018	UCPD_ICR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FRSEVTCF	Res	Res	Res	Res	TYPECEVT2CF	TYPECEVT1CF	Res	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res	Res	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res
	Reset value												0					0	0		0	0	0	0	0			0	0	0	0	0	0
0x01C	UCPD_TX_ORDSETR	Res	Res	Res	Res	Res	Res	Res	Res	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																																
0x020	UCPD_TX_PAYSZR	Res	Res	Res	Res	Res	Res	Res	Res	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	UCPD_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	Res
	Reset value																																
0x028	UCPD_RX_ORDSETR	Res	Res	Res	Res	Res	Res	Res	Res	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																																
0x02C	UCPD_RX_PAYSZR	Res	Res	Res	Res	Res	Res	Res	Res	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	UCPD_RXDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
	Reset value																																
0x034	UCPD_RX_ORDEXTR1	Res	Res	Res	Res	Res	Res	Res	Res	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 252. UCPD 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											
0x038	UCPD_RX_ORDEXTR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RXSOPX2[19:0]														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										
0x03C - 0x3FF	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	Res	Res	Res									

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

## 39 HDMI-CEC controller (CEC)

### 39.1 HDMI-CEC introduction

Consumer electronics control (CEC) is part of HDMI (high-definition multimedia interface) standard. It contains a protocol that provides high-level control functions between various audiovisual products. CEC operates at low speeds, with minimum processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

### 39.2 HDMI-CEC controller main features

- Complies with HDMI-CEC v1.4 specification
- 32 kHz CEC kernel with 2 clock source options
  - HSI RC oscillator with fixed prescaler (HSI/488)
  - LSE oscillator
- Works in Stop mode for ultra-low-power applications
- Configurable signal-free time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable peripheral address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration lost detection (ARBLST)
  - with automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)

### 39.3 HDMI-CEC functional description

#### 39.3.1 HDMI-CEC pin

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up must be added externally to the microcontroller.

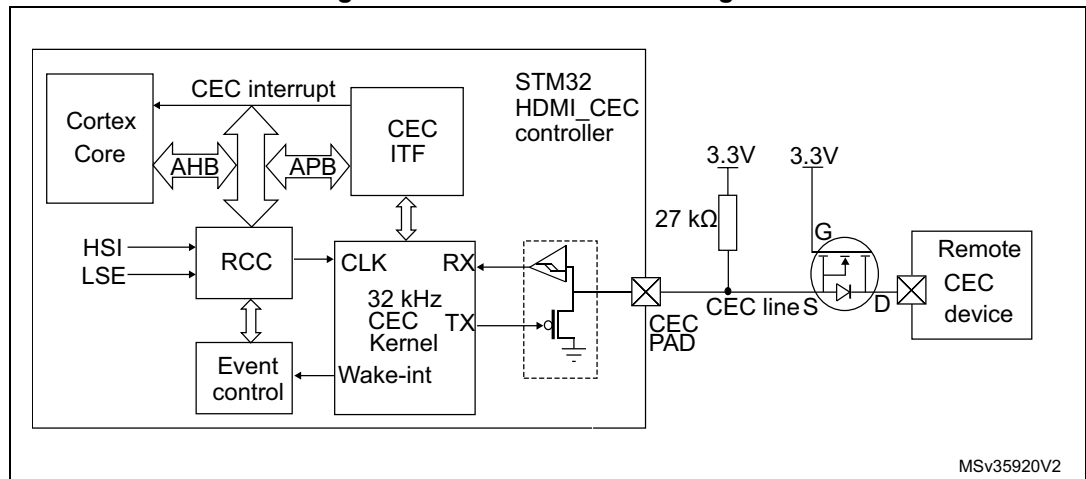
To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This can be done by using a MOS transistor, as shown on [Figure 415](#).

Table 253. HDMI pin

Name	Signal type	Remarks
CEC	Bidirectional	Two states: – 1 = high impedance – 0 = low impedance A 27 kΩ resistor must be added externally.

#### 39.3.2 HDMI-CEC block diagram

Figure 415. HDMI-CEC block diagram



#### 39.3.3 Message description

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgment bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.

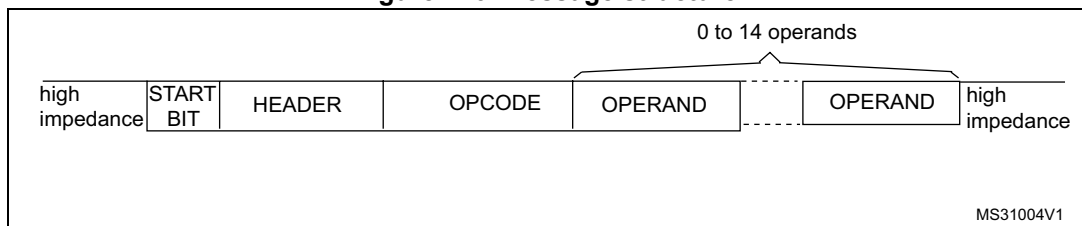
All these blocks are made of a 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In case a message contains additional blocks after an EOM is indicated, those additional blocks must be ignored. The EOM bit may be set in the header block to 'ping' other devices, to make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header, or by the follower that needs to reject a broadcast message.

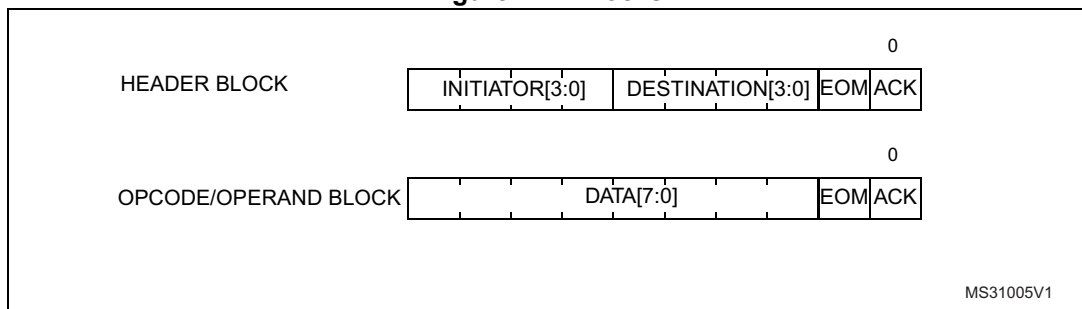
The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

**Figure 416. Message structure**



MS31004V1

**Figure 417. Blocks**



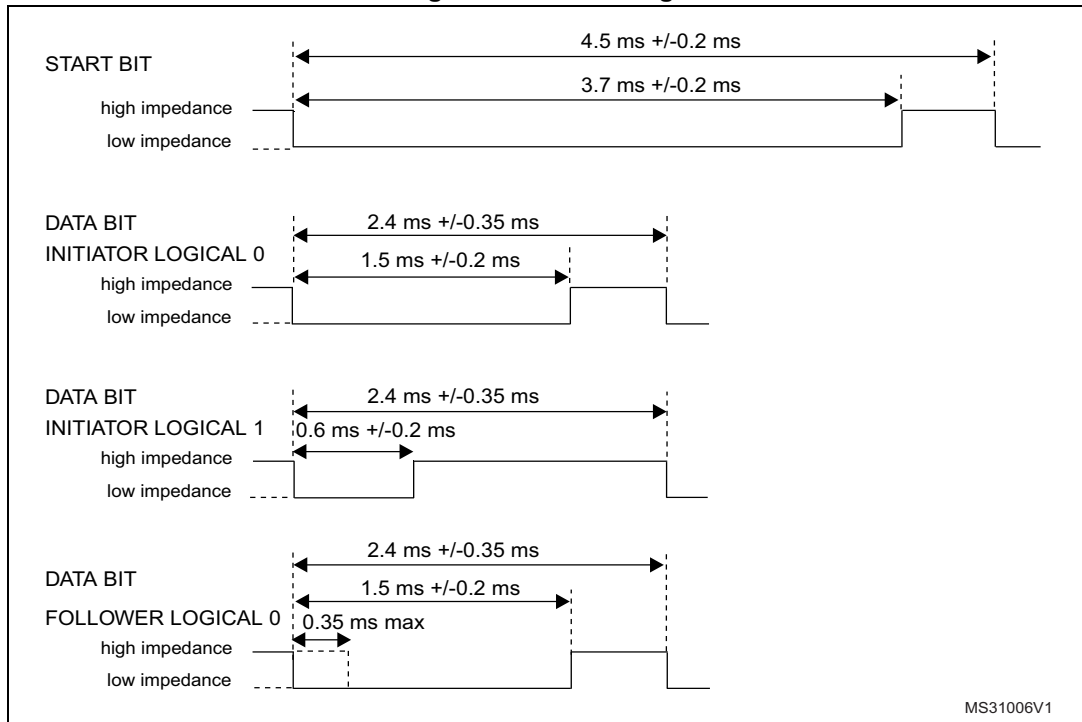
MS31005V1

### 39.3.4 Bit timing

The format of the start bit is unique and identifies the start of a message. It must be validated by its low duration and its total duration.

All remaining data bits in the message, after the start bit, have consistent timing. The high-to-low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.

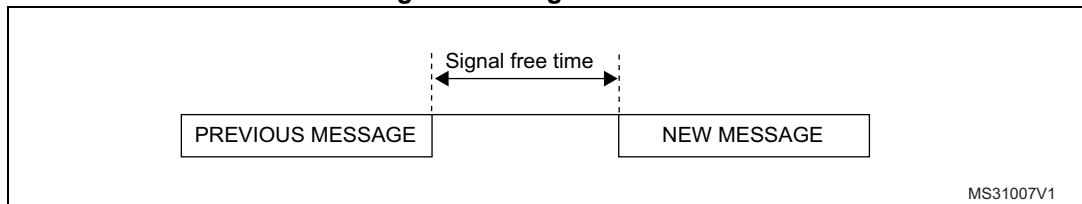
Figure 418. Bit timings



### 39.4 Arbitration

All devices transmitting - or retransmitting - a message onto the CEC line must ensure that it has been inactive for a number of bit periods. This signal-free time is defined as the time starting from the final bit of the previous frame and depends on the initiating device and the current status as shown in the figure below.

Figure 419. Signal free time



Since only one initiator is allowed at any one time, an arbitration mechanism is provided to avoid conflict when more than one initiator begins transmitting at the same time.

CEC line arbitration starts with the leading edge of the start bit and continues until the end of the initiator address bits within the header block. During this period, the initiator must monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0. Assuming then it has lost arbitration, it stops transmitting and becomes a follower.

Figure 420. Arbitration phase

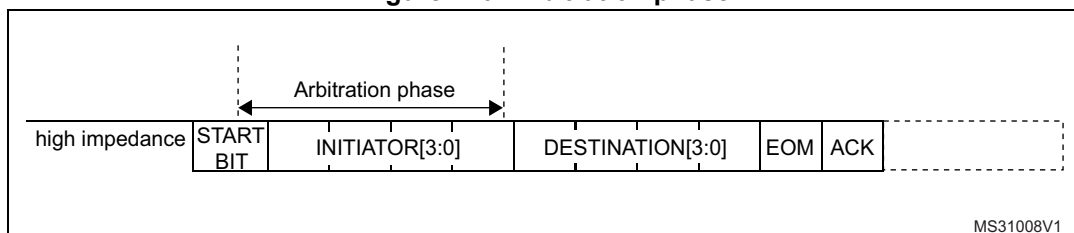
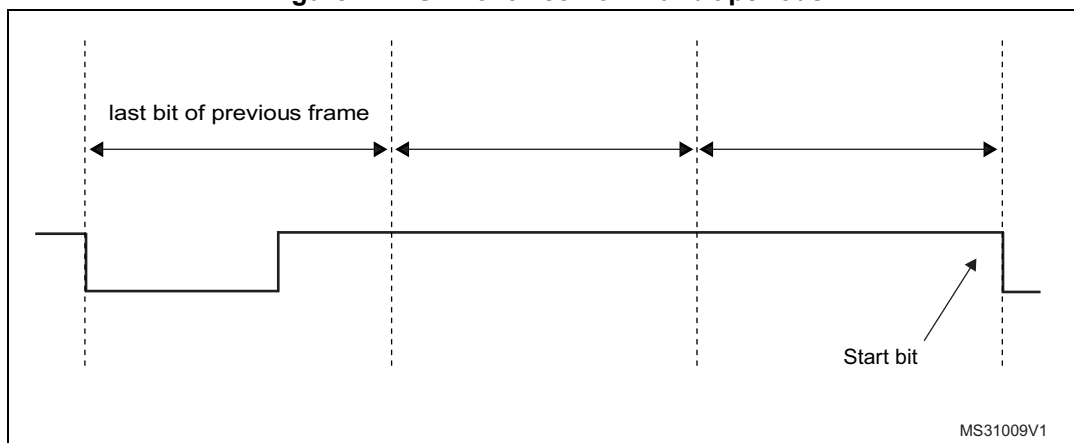


Figure 421 shows an example for a SFT of three nominal bit periods.

Figure 421. SFT of three nominal bit periods



A configurable time window is counted before starting the transmission.

In the SFT = 0 configuration, HDMI-CEC performs automatic SFT calculation ensuring compliance with the HDMI-CEC standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission

This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFT bits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

### 39.4.1 SFT option bit

In case of SFTOPT = 0 configuration, SFT starts being counted when the start-of-transmission command is set by software (TXSOM = 1).

In case of SFTOPT = 1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT timer is still running instead, the system waits until the timer elapses before transmission can start.

In case of SFTOPT = 1 a bus-event condition starting the SFT timer is detected in the following cases:

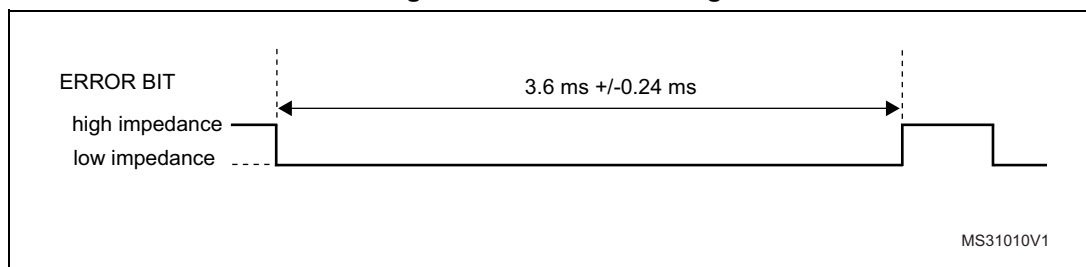
- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR = 1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACHE bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any uncodified low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

## 39.5 Error handling

### 39.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period (3.6 ms nominally).

Figure 422. Error bit timing



### 39.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

### 39.5.3 Bit rising error (BRE)

BRE (bit rising error) is set when a bit rising edge is detected outside the windows where it is expected (see [Figure 423](#)). BRE flag also generates a CEC interrupt if the BREIE = 1.



In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP = 1 an error bit is generated even if BREGEN = 0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN = 0, BRDNOGEN = 1.

### 39.5.4 Short bit period error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see [Figure 423](#)). SBPE flag also generates a CEC interrupt if the SBPEIE = 1.

An error bit is always generated on the line in case of a SBPE error detection. An error bit is not generated upon SBPE detection only when Listen mode is set (LSTN = 1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN = 1

### 39.5.5 Long bit period error (LBPE)

LBPE is set when a bit falling edge is not detected in a valid window (see [Figure 423](#)). LBPE flag also generates a CEC interrupt if the LBPEIE = 1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN = 0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN = 0, BRDNOGEN = 1.

*Note: The BREGEN = 1, BRESTP = 0 configuration must be avoided.*

**Figure 423. Error handling**

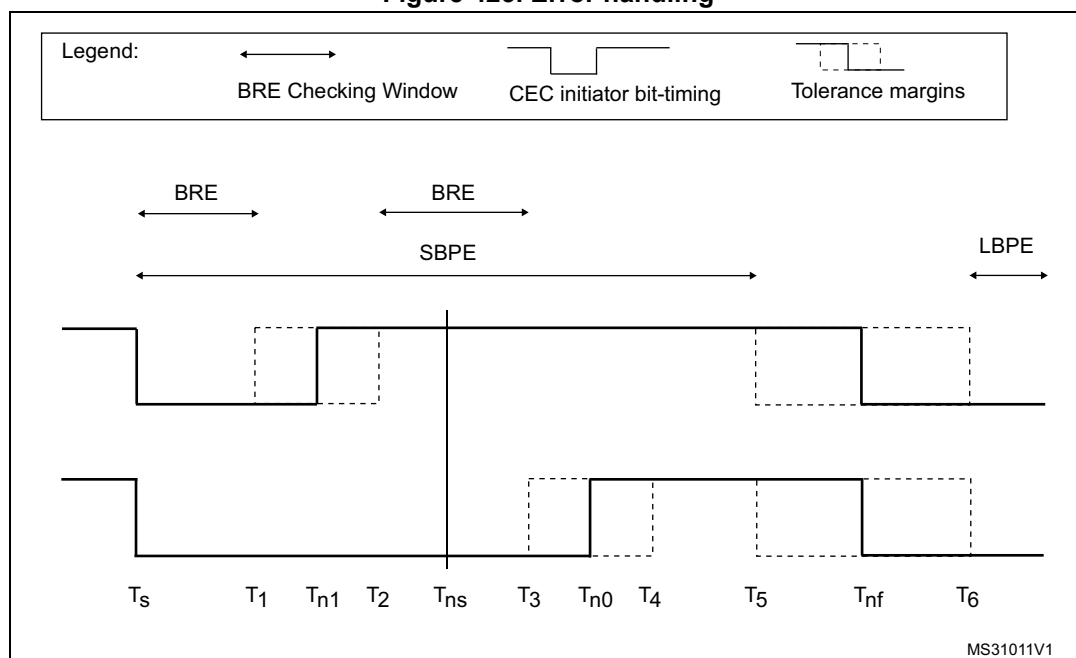


Table 254. Error handling timing parameters

Time	RXTOL	ms	Description
$T_s$	x	0	Bit start event.
$T_1$	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
$T_{n1}$	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
$T_2$	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
$T_{ns}$	x	1.05	Nominal sampling time.
$T_3$	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	
$T_{n0}$	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
$T_4$	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
$T_5$	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
$T_{nf}$	x	2.4	The nominal data bit period.
$T_6$	0	2.75	The latest time for the start of a following bit.
	1	2.95	

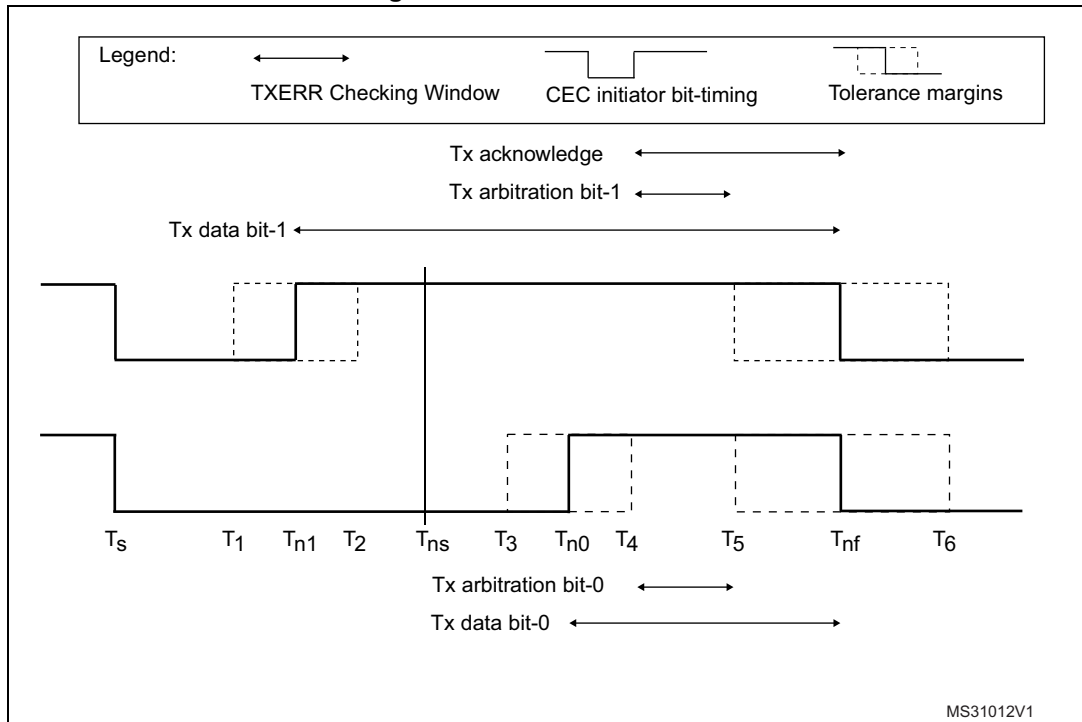
### 39.5.6 Transmission error detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE = 1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to five times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

Figure 424. TXERR detection



MS31012V1

Table 255. TXERR timing parameters

Time	RXTOL	ms	Description
T <sub>s</sub>	x	0	Bit start event.
T <sub>1</sub>	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
T <sub>n1</sub>	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
T <sub>2</sub>	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
T <sub>ns</sub>	x	1.05	Nominal sampling time.
T <sub>3</sub>	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	
T <sub>n0</sub>	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
T <sub>4</sub>	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T <sub>5</sub>	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T <sub>nf</sub>	x	2.4	The nominal data bit period.

Table 255. TXERR timing parameters (continued)

Time	RXTOL	ms	Description
T <sub>6</sub>	0	2.75	The latest time for the start of a following bit.
	1	2.95	

## 39.6 HDMI-CEC interrupts

An interrupt can be produced:

- during reception if a receive block transfer is finished or if a receive error occurs.
- during transmission if a transmit block transfer is finished or if a transmit error occurs.

Table 256. HDMI-CEC interrupts

Interrupt event	Event flag	Enable control bit
Rx-byte received	RXBR	RXBRIE
End of reception	RXEND	RXENDIE
Rx-overflow	RXOVR	RXOVRIE
Rxbit rising error	BRE	BREIE
Rx-short bit period error	SBPE	SBPEIE
Rx-long bit period error	LBPE	LBPEIE
Rx-missing acknowledge error	RXACKE	RXACKEIE
Arbitration lost	ARBLST	ARBLSTIE
Tx-byte request	TXBR	TXBRIE
End of transmission	TXEND	TXENDIE
Tx-buffer underrun	TXUDR	TXUDRIE
Tx-error	TXERR	TXERRIE
Tx-missing acknowledge error	TXACKE	TXACKEIE

## 39.7 HDMI-CEC registers

Refer to [Section 1.2](#) for a list of abbreviations used in register descriptions. The registers have to be accessed by words (32 bits).

### 39.7.1 CEC control register (CEC\_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	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.	TXEOM	TXSOM	CECEN
													rs	rs	rw

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

#### Bit 2 TXEOM: Tx end of message

The TXEOM bit is set by software to command transmission of the last byte of a CEC message. TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.  
 0: TXDR data byte is transmitted with EOM = 0.  
 1: TXDR data byte is transmitted with EOM = 1.

*Note: TXEOM must be set when CECEN = 1.*

*TXEOM must be set before writing transmission data to TXDR.*

*If TXEOM is set when TXSOM = 0, transmitted message consists of 1 byte (HEADER) only (PING message).*

#### Bit 1 TXSOM: Tx start of message

TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM. Start-bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission starts after the end of reception. TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND = 1), in case of transmission underrun (TXUDR = 1), negative acknowledge (TXACKE = 1), and transmission error (TXERR = 1). It is also cleared by CECEN = 0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST = 1). TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.

0: No CEC transmission is on-going.

1: CEC transmission command

*Note: TXSOM must be set when CECEN = 1.*

*TXSOM must be set when transmission data is available into TXDR.*

*HEADER first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC\_CFGR.OAR that is used only for reception.*

Bit 0 **CECEN**: CEC enable

The CECEN bit is set and cleared by software. CECEN = 1 starts message reception and enables the TXSOM control. CECEN = 0 disables the CEC peripheral, clears all bits of CEC\_CR register and aborts any on-going reception or transmission.

- 0: CEC peripheral is off.
- 1: CEC peripheral is on.

### 39.7.2 CEC configuration register (CEC\_CFGR)

This register is used to configure the HDMI-CEC controller.

Address offset: 0x04

Reset value: 0x0000 0000

**Caution:** It is mandatory to write CEC\_CFGR only when CECEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSTN		OAR[14:0]													
r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFTOP	BRDN OGEN	LBPEG EN	BREGE N	BREST P	RXTOL	SFT[2:0]		
							r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit 31 **LSTN**: Listen mode

LSTN bit is set and cleared by software.

0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.

1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

Bits 30:16 **OAR[14:0]**: Own addresses configuration

The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.

At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN = 1), but without acknowledge sent. Broadcast messages are always received.

Example:

OAR = 0b000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5. Consequently, each message directed to one of these addresses is received.

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

Bit 8 **SFTOP**: SFT option bit

The SFTOPT bit is set and cleared by software.

0: SFT timer starts when TXSOM is set by software.

1: SFT timer starts automatically at the end of message transmission/reception.

- Bit 7 BRDNOGEN:** Avoid error-bit generation in broadcast  
 The BRDNOGEN bit is set and cleared by software.  
 0: BRE detection with BRESTP = 1 and BREGEN = 0 on a broadcast message generates an error-bit on the CEC line. LBPE detection with LBPEGEN = 0 on a broadcast message generates an error-bit on the CEC line.  
 1: Error-bit is not generated in the same condition as above. An error-bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.
- Bit 6 LBPEGEN:** Generate error-bit on long bit period error  
 The LBPEGEN bit is set and cleared by software.  
 0: LBPE detection does not generate an error-bit on the CEC line.  
 1: LBPE detection generates an error-bit on the CEC line.  
*Note: If BRDNOGEN = 0, an error-bit is generated upon LBPE detection in broadcast even if LBPEGEN = 0.*
- Bit 5 BREGEN:** Generate error-bit on bit rising error  
 The BREGEN bit is set and cleared by software.  
 0: BRE detection does not generate an error-bit on the CEC line.  
 1: BRE detection generates an error-bit on the CEC line (if BRESTP is set).  
*Note: If BRDNOGEN = 0, an error-bit is generated upon BRE detection with BRESTP = 1 in broadcast even if BREGEN = 0.*
- Bit 4 BRESTP:** Rx-stop on bit rising error  
 The BRESTP bit is set and cleared by software.  
 0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.  
 1: BRE detection stops message reception.
- Bit 3 RXTOL:** Rx-tolerance  
 The RXTOL bit is set and cleared by software.  
 0: Standard tolerance margin:  
 – Start-bit, +/- 200  $\mu$ s rise, +/- 200  $\mu$ s fall  
 – Data-bit: +/- 200  $\mu$ s rise. +/- 350  $\mu$ s fall  
 1: Extended tolerance  
 – Start-bit: +/- 400  $\mu$ s rise, +/- 400  $\mu$ s fall  
 – Data-bit: +/-300  $\mu$ s rise, +/- 500  $\mu$ s fall
- Bits 2:0 SFT[2:0]:** Signal free time  
 SFT bits are set by software. In the SFT = 0x0 configuration, the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.  
 0x0  
 – 2.5 data-bit periods if CEC is the last bus initiator with unsuccessful transmission (ARBLST = 1, TXERR = 1, TXUDR = 1 or TXACKE = 1)  
 – 4 data-bit periods if CEC is the new bus initiator  
 – 6 data-bit periods if CEC is the last bus initiator with successful transmission (TXEOM = 1)  
 0x1: 0.5 nominal data bit periods  
 0x2: 1.5 nominal data bit periods  
 0x3: 2.5 nominal data bit periods  
 0x4: 3.5 nominal data bit periods  
 0x5: 4.5 nominal data bit periods  
 0x6: 5.5 nominal data bit periods  
 0x7: 6.5 nominal data bit periods

### 39.7.3 CEC Tx data register (CEC\_TXDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	TXD[7:0]							
								w	w	w	w	w	w	w	w

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

Bits 7:0 **TXD[7:0]**: Tx data

TXD is a write-only register containing the data byte to be transmitted.

### 39.7.4 CEC Rx data register (CEC\_RXDR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	RXD[7:0]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **RXD[7:0]**: Rx data

RXD is read-only and contains the last data byte that has been received from the CEC line.

### 39.7.5 CEC interrupt and status register (CEC\_ISR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACK E	TXERR	TXUDR	TXEND	TXBR	ARBLS T	RXACK E	LBPE	SBPE	BRE	RXOV R	RXEND	RXBR
			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.



- Bit 12 **TXACKE**: Tx-missing acknowledge error  
In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.  
TXACKE is cleared by software write at 1.
- Bit 11 **TXERR**: Tx-error  
In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.  
TXERR is cleared by software write at 1.
- Bit 10 **TXUDR**: Tx-buffer underrun  
In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.  
TXUDR is cleared by software write at 1
- Bit 9 **TXEND**: End of transmission  
TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.  
TXEND is cleared by software write at 1.
- Bit 8 **TXBR**: Tx-byte request  
TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within six nominal data-bit periods before transmission underrun error occurs (TXUDR).  
TXBR is cleared by software write at 1.
- Bit 7 **ARBLST**: Arbitration lost  
ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.  
ARBLST is cleared by software write at 1.
- Bit 6 **RXACKE**: Rx-missing acknowledge  
In receive mode, RXACKE is set by hardware to inform application that no acknowledge was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception.  
RXACKE is cleared by software write at 1.
- Bit 5 **LBPE**: Rx-long bit period error  
LBPE is set by hardware in case a data-bit waveform is detected with long bit period error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an error-bit on the CEC line if LBPEGEN = 1. In case of broadcast, error-bit is generated even in case of LBPEGEN = 0.  
LBPE is cleared by software write at 1.
- Bit 4 **SBPE**: Rx-short bit period error  
SBPE is set by hardware in case a data-bit waveform is detected with short bit period error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an error-bit on the CEC line.  
SBPE is cleared by software write at 1.

Bit 3 **BRE**: Rx-bit rising error

BRE is set by hardware in case a data-bit waveform is detected with bit rising error. BRE is set either at the time the misplaced rising edge occurs, or at the end of the maximum BRE tolerance allowed by RXTOL, in case rising edge is still longing. BRE stops message reception if BRESTP = 1. BRE generates an error-bit on the CEC line if BREGEN = 1.

BRE is cleared by software write at 1.

Bit 2 **RXOVR**: Rx-overflow

RXOVR is set by hardware if RXBR is not yet cleared at the time a new byte is received on the CEC line and stored into RXD. RXOVR assertion stops message reception so that no acknowledge is sent. In case of broadcast, a negative acknowledge is sent.

RXOVR is cleared by software write at 1.

Bit 1 **RXEND**: End of reception

RXEND is set by hardware to inform application that the last byte of a CEC message is received from the CEC line and stored into the RXD buffer. RXEND is set at the same time of RXBR.

RXEND is cleared by software write at 1.

Bit 0 **RXBR**: Rx-byte received

The RXBR bit is set by hardware to inform application that a new byte has been received from the CEC line and stored into the RXD buffer.

RXBR is cleared by software write at 1.

### 39.7.6 CEC interrupt enable register (CEC\_IER)

Address offset: 0x14

Reset value: 0x0000 0000

**Caution:** It is mandatory to write CEC\_IER only when CECEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACK IE	TXERR IE	TXUDR IE	TXEND IE	TXBRI E	ARBL S TIE	RXACK IE	LBPEI E	SBPEI E	BREIE	RXOV RIE	RXEND IE	RXBRI E
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKIE**: Tx-missing acknowledge error interrupt enable

The TXACKEIE bit is set and cleared by software.

0: TXACKE interrupt disabled

1: TXACKE interrupt enabled

Bit 11 **TXERRIE**: Tx-error interrupt enable

The TXERRIE bit is set and cleared by software.

0: TXERR interrupt disabled

1: TXERR interrupt enabled

Bit 10 **TXUDRIE**: Tx-underrun interrupt enable

The TXUDRIE bit is set and cleared by software.

0: TXUDR interrupt disabled

1: TXUDR interrupt enabled

- Bit 9 **TXENDIE**: Tx-end of message interrupt enable  
The TXENDIE bit is set and cleared by software.  
0: TXEND interrupt disabled  
1: TXEND interrupt enabled
- Bit 8 **TXBRIE**: Tx-byte request interrupt enable  
The TXBRIE bit is set and cleared by software.  
0: TXBR interrupt disabled  
1: TXBR interrupt enabled
- Bit 7 **ARBLSTIE**: Arbitration lost interrupt enable  
The ARBLSTIE bit is set and cleared by software.  
0: ARBLST interrupt disabled  
1: ARBLST interrupt enabled
- Bit 6 **RXACKIE**: Rx-missing acknowledge error interrupt enable  
The RXACKIE bit is set and cleared by software.  
0: RXACKE interrupt disabled  
1: RXACKE interrupt enabled
- Bit 5 **LBPEIE**: Long bit period error interrupt enable  
The LBPEIE bit is set and cleared by software.  
0: LBPE interrupt disabled  
1: LBPE interrupt enabled
- Bit 4 **SBPEIE**: Short bit period error interrupt enable  
The SBPEIE bit is set and cleared by software.  
0: SBPE interrupt disabled  
1: SBPE interrupt enabled
- Bit 3 **BREIE**: Bit rising error interrupt enable  
The BREIE bit is set and cleared by software.  
0: BRE interrupt disabled  
1: BRE interrupt enabled
- Bit 2 **RXOVRIE**: Rx-buffer overrun interrupt enable  
The RXOVRIE bit is set and cleared by software.  
0: RXOVR interrupt disabled  
1: RXOVR interrupt enabled
- Bit 1 **RXENDIE**: End of reception interrupt enable  
The RXENDIE bit is set and cleared by software.  
0: RXEND interrupt disabled  
1: RXEND interrupt enabled
- Bit 0 **RXBRIE**: Rx-byte received interrupt enable  
The RXBRIE bit is set and cleared by software.  
0: RXBR interrupt disabled  
1: RXBR interrupt enabled

39.7.7 HDMI-CEC register map

Table 257. HDMI-CEC 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	CEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			
	Reset value																																	0	0	0
0x04	CEC_CFGR	LSTN	OAR[14:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFTOPT	BRDNOGEN	LBPEGEN	BREGEN	BRESTP	RXTOL	SFT[2:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x08	CEC_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]										
	Reset value																										0	0	0	0	0	0	0	0	0	0
0x0C	CEC_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]										
	Reset value																																			
0x10	CEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKIE	TXERR	TXUDR	TXEND	TXBRIE	ARBLSSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVR	RXEND	RXBRIE		
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	CEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVR	RXENDIE	RXBRIE		
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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



## 40 Debug support (DBG)

### 40.1 Overview

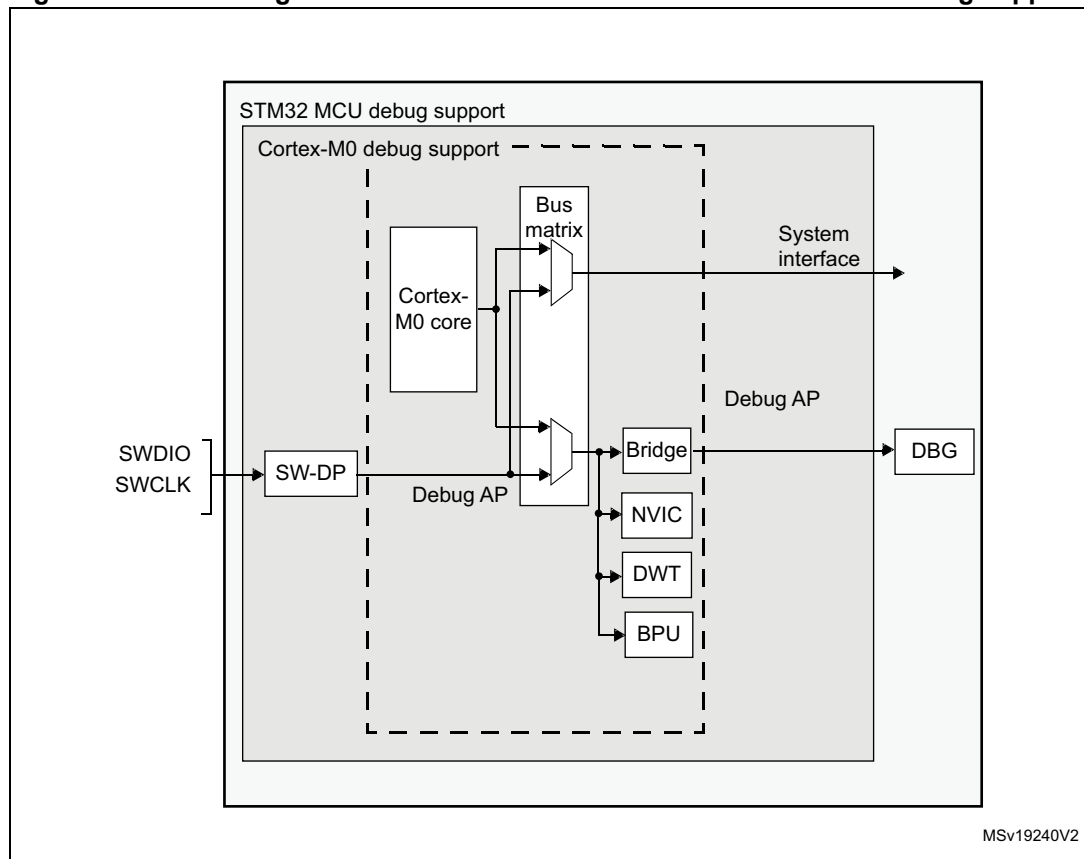
The STM32G0x1 devices are built around a Cortex<sup>®</sup>-M0+ core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state 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 STM32G0x1 MCUs.

One interface for debug is available:

- Serial wire

**Figure 425. Block diagram of STM32G0x1 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 STM32G0x1:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

*Note:* For further information on debug functionality supported by the Arm Cortex<sup>®</sup>-M0+ core, refer to the Cortex<sup>®</sup>-M0+ Technical Reference Manual (see [Section 40.2: Reference Arm documentation](#)).

## 40.2 Reference Arm documentation

- Cortex<sup>®</sup>-M0+ Technical Reference Manual (TRM), available from <http://infocenter.arm.com>
- Arm Debug Interface V5
- Arm CoreSight Design Kit revision r1p1 Technical Reference Manual

## 40.3 Pinout and debug port pins

The STM32G0x1 MCUs are available in various packages with different numbers of available pins.

### 40.3.1 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 258. 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

### 40.3.2 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 7.3.2: I/O pin alternate function multiplexer and mapping on page 230](#).

### 40.3.3 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.*

## 40.4 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).

## 40.5 SWD port

### 40.5.1 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 target board (100 kΩ recommended by Arm). Having embedded pull-up on SWDIO line removes the need to add external resistor.

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.

### 40.5.2 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 259. 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 263 on page 1358</a> )
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as "1" by the target because of the pull-up

Refer to the Cortex<sup>®</sup>-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 260. 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 261. 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.

### 40.5.3 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<sup>®</sup>-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<sup>®</sup>-M0+ TRM* and the *CoreSight Design Kit r1p0 TRM*.

#### 40.5.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result. The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)  
This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it fails.

#### 40.5.5 SW-DP registers

Access to these registers are initiated when APnDP=0

**Table 262. 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 <sup>®</sup> -M0+: <b>0x0BC11477</b> (identifies the SW-DP)
00	Write		ABORT	

**Table 262. SW-DP registers (continued)**

A[3:2]	R/W	CTRLSEL bit of SELECT register	Register	Notes
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read		READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write		SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write		READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

**40.5.6 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 263. 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-up – Configure the transfer operation for AP accesses – Control the pushed compare and pushed verify operations. – Read some status flags (overrun, power-up acknowledges)

**Table 263. 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)

## 40.6 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 264. 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<sup>®</sup>-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

## 40.7 BPU (Break Point Unit)

The Cortex<sup>®</sup>-M0+ BPU implementation provides four breakpoint registers. The BPU is a subset of the flash memory patch and breakpoint (FPB) block available in Armv7-M (Cortex-M3 & Cortex-M4).

### 40.7.1 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.

## 40.8 DWT (Data Watchpoint)

The Cortex<sup>®</sup>-M0+ DWT implementation provides two watchpoint register sets.

### 40.8.1 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*.

### 40.8.2 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<sup>®</sup>-M0+ DWT\_PCSR records both instructions that pass their condition codes and those that fail.

## 40.9 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

### 40.9.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode: FCLK and HCLK are still active. Consequently, this mode does not impose any restrictions on the standard debug features.
- In Stop/Standby mode, the DBG\_STOP bit must be previously set by the debugger.

This enables the internal RC oscillator clock to feed FCLK and HCLK in Stop mode.

To save power, in Sleep mode, the bus matrix clock is disabled in the RCC if no master is on. As a consequence, if both DMA1 and DMA2 are disabled, the debugger read access to SRAM during Sleep mode is not possible (returns 0x0).

### 40.9.2 Debug support for timers, watchdog and I<sup>2</sup>C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I<sup>2</sup>C, the user can choose to block the SMBUS timeout during a breakpoint.

## 40.10 DBG registers

### 40.10.1 DBG device ID code register (DBG\_IDCODE)

The STM32G0x1 products integrate a device ID code identifying the device and its die revision.

This code is accessible by the software debug port (two pins) or by the user software.

Address offset: 0x00

Reset value: 0xFFFF 64XX (see [Table 265](#))

Only 32-bit access supported. Read-only

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 265](#).

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

These bits are read as 0b0110.

Bits 11:0 **DEV\_ID[11:0]**: Device identifier

This field indicates the device ID. Refer to [Table 265](#).

**Table 265. DEV\_ID and REV\_ID field values**

Device	DEV_ID	Revision code	Revision number	REV_ID
STM32G0B1xx and STM32G0C1xx	0x467	A	1.0	0x1000
		Z, 1, 2	1.1	0x1001

Table 265. DEV\_ID and REV\_ID field values

Device	DEV_ID	Revision code	Revision number	REV_ID
STM32G071xx and STM32G081xx	0x460	A, Z	1.0	0x1000
		B	2.0	0x2000
		Y, 1	2.1	0x2001
STM32G051xx and STM32G061xx	0x456	A	1.0	0x1000
		Z	1.1	0x1001
STM32G031xx and STM32G041xx	0x466	A	1.0	0x1000
		Z	1.1	0x1001
		Y, 3, 4	1.2	0x1003

### 40.10.2 DBG configuration register (DBG\_CR)

This register configures the low-power modes of the MCU under debug.

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support this feature, it is still possible for the user software to write to this register.

Address offset: 0x04

Power-on reset value: 0x0000 0000 (not reset by system reset)

Only 32-bit access supported

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.	DBG_STANDBY	DBG_STOP	Res.
													r/w	r/w	

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.

### 40.10.3 DBG APB freeze register 1 (DBG\_APB\_FZ1)

This register configures the clocking of timers, RTC, IWDG, WWDG, and I2C SMBUS peripherals of the MCU under debug:

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address offset: 0x08

Power-on reset value: 0x0000 0000 (not reset by system reset)

Only 32-bit access are supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_LPTIM1_STOP	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C2_SMBUS_TIMEOUT <sup>(1)</sup>	DBG_I2C1_SMBUS_TIMEOUT	Res.	Res.	Res.	Res.	Res.
rw	rw								rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	DBG_TIM4_STOP <sup>(2)</sup>	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw	rw					rw	rw		rw	rw	rw



1. Only significant on devices integrating I2C2, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).
2. Only significant on devices integrating TIM4, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#).

Bit 31 **DBG\_LPTIM1\_STOP**: Clocking of LPTIMER1 counter when the core is halted  
This bit enables/disables the clock to the counter of LPTIMER1 when the core is halted:  
0: Enable  
1: Disable

Bit 30 **DBG\_LPTIM2\_STOP**: Clocking of LPTIMER2 counter when the core is halted  
This bit enables/disables the clock to the counter of LPTIMER2 when the core is halted:  
0: Enable  
1: Disable

Bits 29:23 Reserved, must be kept at reset value.

Bit 22 **DBG\_I2C2\_SMBUS\_TIMEOUT**: SMBUS timeout when core is halted  
0: Same behavior as in normal mode  
1: The SMBUS timeout is frozen

Bit 21 **DBG\_I2C1\_SMBUS\_TIMEOUT**: SMBUS timeout 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 9:6 Reserved, must be kept at reset value.

Bit 5 **DBG\_TIM7\_STOP**: Clocking of TIM7 counter when the core is halted.  
This bit enables/disables the clock to the counter of ITIM7 when the core is halted:  
0: Enable  
1: Disable

Bit 4 **DBG\_TIM6\_STOP**: Clocking of TIM6 counter when the core is halted  
This bit enables/disables the clock to the counter of TIM6 when the core is halted:  
0: Enable  
1: Disable

Bit 3 Reserved, must be kept at reset value.



- Bit 2 **DBG\_TIM4\_STOP**: Clocking of TIM4 counter when the core is halted  
 This bit enables/disables the clock to the counter of TIM4 when the core is halted:  
 0: Enable  
 1: Disable
- Bit 1 **DBG\_TIM3\_STOP**: Clocking of TIM3 counter when the core is halted  
 This bit enables/disables the clock to the counter of TIM3 when the core is halted:  
 0: Enable  
 1: Disable
- Bit 0 **DBG\_TIM2\_STOP**: Clocking of TIM2 counter when the core is halted  
 This bit enables/disables the clock to the counter of TIM2 when the core is halted:  
 0: Enable  
 1: Disable

#### 40.10.4 DBG APB freeze register 2 (DBG\_APB\_FZ2)

This register configures the clocking of timer counters when the MCU is under debug. It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address offset: 0x0C

Power-on reset value: 0x0000 0000 (not reset by system reset)

Only 32-bit access is supported.

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.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP <sup>(1)</sup>
													r/w	r/w	r/w
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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r/w				r/w											

1. Only significant on devices integrating TIM15, otherwise reserved. Refer to [Section 1.4: Availability of peripherals](#)

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

Bit 18 **DBG\_TIM17\_STOP**: Clocking of TIM17 counter when the core is halted  
 This bit enables/disables the clock to the counter of TIM17 when the core is halted:  
 0: Enable  
 1: Disable

Bit 17 **DBG\_TIM16\_STOP**: Clocking of TIM16 counter when the core is halted  
 This bit enables/disables the clock to the counter of TIM16 when the core is halted:  
 0: Enable  
 1: Disable

Bit 16 **DBG\_TIM15\_STOP**: Clocking of TIM15 counter when the core is halted  
 This bit enables/disables the clock to the counter of TIM15 when the core is halted:  
 0: Enable  
 1: Disable

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.

### 40.10.5 DBG register map

The following table summarizes the Debug registers.

**Table 266. DBG register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	<b>DBG_IDCODE</b>	REV_ID																Res.	Res.	Res.	Res.	DEV_ID											
	Reset value <sup>(1)</sup>	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X
0x04	<b>DBG_CR</b>	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																															0	0

Table 266. DBG register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	DBG_APB_FZ1	DBG_LPTIM1_STOP	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	Res.	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
		Reset value	0	0								0	0									0	0	0					0	0		0	0
0x0C	DBG_APB_FZ2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	DBG_TIM14_STOP	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		Reset value														0	0	0	0				0										

1. The reset value is product dependent. For more information, refer to [Section 40.10.1: DBG device ID code register \(DBG\\_IDCODE\)](#).

Refer to [Section 2.2 on page 60](#) for the register boundary addresses.

# 41 Device electronic signature

The device electronic signature is stored in the System memory area of the flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the STM32G0x1 microcontroller.

## 41.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

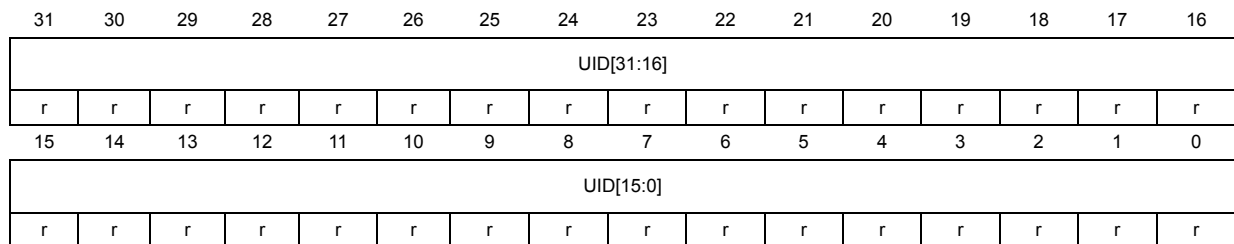
- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x1FFF 7590

Address offset: 0x00

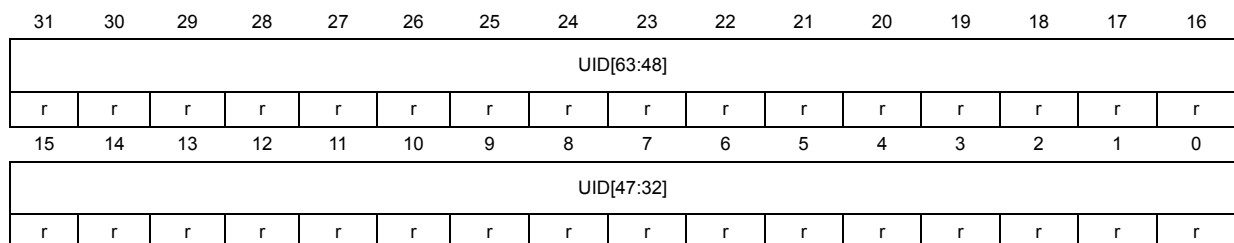
Read only = 0xXXXX XXXX where X is factory-programmed



Bits 31:0 **UID[31:0]**: X and Y coordinates on the wafer expressed in BCD format

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

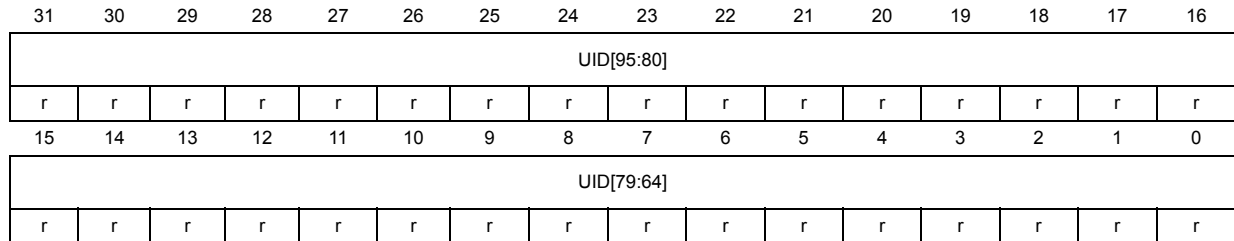


Bits 31:8 **UID[63:40]:** LOT\_NUM[23:0]  
 Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF\_NUM[7:0]  
 Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed



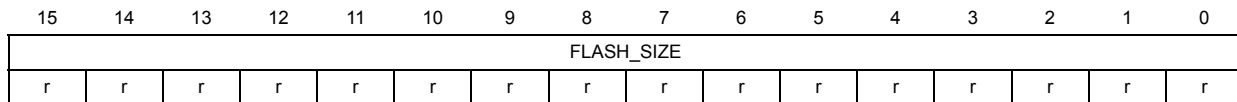
Bits 31:0 **UID[95:64]:** LOT\_NUM[55:24]  
 Lot number (ASCII encoded)

## 41.2 Flash memory size data register

Base address: 0x1FFF 75E0

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed



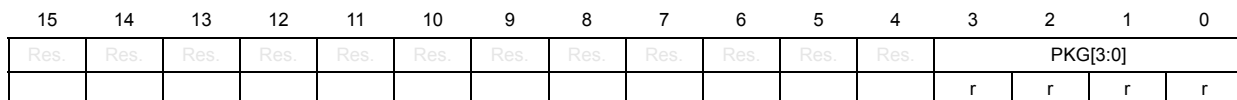
Bits 15:0 **FLASH\_SIZE[15:0]:** Flash memory size  
 This bitfield indicates the size of the device Flash memory expressed in Kbytes.  
 As an example, 0x040 corresponds to 64 Kbytes.

## 41.3 Package data register

Base address: 0x1FFF 7500

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed



Bits 15:4 Reserved

Bits 3:0 **PKG[3:0]**: Package type

**Condition: STM32G071xx and STM32G081xx**

0000: UFQFPN28 GP version

0001: UFQFPN28 N version

0100: UFQFPN32 / LQFP32 GP version

0101: UFQFPN32 / LQFP32 N version

1000: UFQFPN48 / LQFP48

1100: LQPF64

Others: Reserved

**Condition: STM32G031xx, STM32G041xx, STM32G051xx, and STM32G061xx**

0001: SO8

0010: WLCSP18

0011: TSSOP20

0100: UFQFPN28

0101: UFQFPN32 / LQFP32

0111: UFQFPN48 / LQFP48

Others: Reserved

**Condition: STM32G0B1xx and STM32G0C1xx**

0000: LQFP100

0001: LQFP32/UFQFPN32 GP version

0010: LQFP32/UFQFPN32 N version

0100: LQFP48 / UFQFPN48 GP version

0101: LQFP48 / UFQFPN48 N version

0110: WLCSP52

0111: LQFP64 GP version

1000: LQFP64 N version

1010: UFBGA64 N version

1011: LQFP80

1100: UFBGA100

Others: Reserved

## 42 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.

## 43 Revision history

**Table 267. Document revision history**

Date	Revision	Changes
29-Oct-2018	1	Initial release.
17-Apr-2019	2	Integration of STM32G031xx and STM32G041xx, affecting: <ul style="list-style-type: none"> <li>– Section <i>Availability of peripherals</i></li> <li>– Figure <i>Memory map</i></li> <li>– Table <i>STM32G031xx and STM32G041xx memory boundary addresses</i> (added)</li> <li>– Section <i>Embedded SRAM</i></li> <li>– Section <i>Boot configuration</i></li> <li>– Section <i>FLASH functional description</i></li> <li>– Table <i>Flash memory organization</i> (title modified)</li> <li>– Section <i>Power control (PWR)</i> (indication of bits not available on STM32G031xx and STM32G041xx)</li> <li>– Figure <i>Clock tree</i></li> <li>– Section <i>Reset and clock control (RCC)</i> (indication of bits not available on STM32G031xx and STM32G041xx) and Section <i>Timer clock</i></li> <li>– Section <i>System configuration controller (SYSCFG)</i> (indication of bits not available on STM32G031xx and STM32G041xx)</li> <li>– Section <i>SYSCFG configuration register 2 (SYSCFG_CFGR2)</i> (clamping diode enable bits added)</li> <li>– Section <i>Introduction</i></li> <li>– Table <i>DMA implementation</i></li> <li>– Table <i>DMAMUX instantiation</i></li> <li>– Section <i>Interrupt and exception vectors</i></li> <li>– Section <i>Extended interrupt and event controller (EXTI)</i> (indication of bits not available on STM32G031xx and STM32G041xx)</li> <li>– Figure <i>General-purpose timer block diagram</i> and Figure <i>External trigger input block</i></li> <li>– Section <i>Infrared interface (IRTIM)</i></li> <li>– Table <i>USART features</i></li> <li>– Table <i>DEV_ID and REV_ID field values</i></li> <li>– Table <i>DBG APB freeze register 2 (DBG_APB_FZ2)</i></li> <li>– Section <i>Package data register</i></li> </ul>



Table 267. Document revision history (continued)

Date	Revision	Changes
19-May-2020	3	<ul style="list-style-type: none"> <li>– <i>Empty check</i> section</li> <li>– <i>Section 3.3.8: FLASH main memory programming sequences</i></li> <li>– <i>User and read protection option bytes</i> section</li> <li>– <i>Option byte loading (OBL)</i> section</li> <li>– <i>Table 20: Access status versus protection level and execution modes</i></li> <li>– <i>Section 3.5.4: Securable memory area</i></li> <li>– <i>Section 3.7.1: FLASH access control register (FLASH_ACR)</i></li> <li>– <i>Section 3.7.8: FLASH option register (FLASH_OPTR)</i> (BORR_LEV[1:0] swapped with BORF_LEV[1:0])</li> <li>– <i>Section 3.7.9: FLASH PCROP area A start address register (FLASH_PCROP1ASR) to Section 3.7.14: FLASH PCROP area B end address register (FLASH_PCROP1BER)</i>: reset values</li> <li>– <i>Section 7.3: GPIO functional description</i>: introductory information modified</li> <li>– <i>Section 7.3.16: USB PD / Dead battery support</i>: description filled</li> <li>– <i>Table 51: Programmable data width and endian behavior (when PINC = MINC = 1)</i>: NDT in the first row corrected from 8 to 4</li> <li>– <i>Table 55: DMAMUX: assignment of multiplexer inputs to resources</i>: TIM16/17_TRG_COM corrected to TIM16/17_COM</li> <li>– <i>Section 15.2: ADC main features</i>: <math>V_{TS}</math> corrected to <math>V_{SENSE}</math></li> <li>– <i>Section 15.3.1: ADC pins and internal signals</i>: tables and their organization (<i>External triggers</i> table brought to this section)</li> <li>– <i>Table 74: Latency between trigger and start of conversion</i>: latency values</li> <li>– <i>Section : Calculating the actual <math>V_{REF+}</math> voltage using the internal reference voltage</i> - corrected from <math>V_{DDA}</math> to <math>V_{REF+}</math></li> <li>– <i>Section 20: AES hardware accelerator (AES)</i>: general update</li> <li>– <i>Section 21: Advanced-control timer (TIM1)</i>: general update</li> <li>– <i>Figure 187: Capture/Compare channel 1 main circuit</i> and <i>Figure 188: Output stage of Capture/Compare channel (channel 1)</i> updated</li> <li>– <i>Figure 205: Master/slave connection example with 1 channel only timers</i> added</li> <li>– <i>Table 124: Output control bit for standard OCx channels</i> updated</li> <li>– <i>Section 22.4.29: TIM3 timer input selection register (TIM3_TISEL)</i>: removed TI4SEL[3:0] and TI3SEL[3:0]</li> <li>– <i>Figure 221: General-purpose timer block diagram (TIM14)</i>: updated</li> <li>– <i>Figure 232: Capture/compare channel 1 main circuit</i> and <i>Figure 233: Output stage of capture/compare channel (channel 1)</i> updated</li> <li>– <i>Section 24.3.11: Using timer output as trigger for other timers (TIM14)</i> added</li> <li>– <i>Figure 251: Capture/compare channel 1 main circuit</i> updated</li> <li>– <i>Section 25.4.24: Using timer output as trigger for other timers (TIM16/TIM17)</i> added</li> <li>– Former <i>Section 28.3.4 Advanced watchdog interrupt feature</i> moved to <i>Section 29.4: WWDG interrupts</i></li> <li>– <i>Section 32.4.2: I2C pins and internal signals</i> added</li> <li>– <i>Section 32.9.3: I2C own address 1 register (I2C_OAR1)</i> and <i>Section 32.9.8: I2C interrupt clear register (I2C_ICR)</i> updated</li> </ul>

Table 267. Document revision history (continued)

Date	Revision	Changes
19-May-2020	3 cont'd	<ul style="list-style-type: none"> <li>– <a href="#">Section 33.4: USART implementation</a> updated - tables reorganized</li> <li>– <a href="#">Section 34.3: LPUART implementation</a> updated - tables reorganized</li> <li>– <a href="#">Section 38: USB Type-C®/USB Power Delivery interface (UCPD)</a>: general update</li> <li>– former <i>DAC trigger connection table</i> renamed as <a href="#">Table 85: DAC interconnection</a> and moved to <a href="#">Section 16.4.2: DAC pins and internal signals</a></li> <li>– <a href="#">Table 265: DEV_ID and REV_ID field values</a></li> <li>– <a href="#">Section 40.10.2: DBG configuration register (DBG_CR)</a></li> </ul>
6-Oct-2020	4	Updated <a href="#">Section 3.4.2: FLASH option byte programming</a> .
20-Nov-2020	5	<p>Extension of the document scope to cover STM32G051xx, STM32G061xx, STM32G0B1xx, and STM32G0C1xx devices, with corresponding addition or update of:</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 1.1: General information</a></li> <li>– <a href="#">Section 1.4: Availability of peripherals</a></li> <li>– <a href="#">Section 2: Memory and bus architecture</a></li> <li>– <a href="#">Section 3: Embedded flash memory (FLASH)</a></li> <li>– <a href="#">Section 4: Power control (PWR)</a></li> <li>– <a href="#">Section 5: Reset and clock control (RCC)</a></li> <li>– <a href="#">Section 8: System configuration controller (SYSCFG)</a></li> <li>– <a href="#">Section 9: Interconnect matrix</a></li> <li>– <a href="#">Section 10: Direct memory access controller (DMA)</a></li> <li>– <a href="#">Section 11: DMA request multiplexer (DMAMUX)</a></li> <li>– <a href="#">Section 12: Nested vectored interrupt controller (NVIC)</a></li> <li>– <a href="#">Section 13: Extended interrupt and event controller (EXTI)</a></li> <li>– <a href="#">Section 15: Analog-to-digital converter (ADC)</a></li> <li>– <a href="#">Section 18: Comparator (COMP)</a></li> <li>– <a href="#">Section 20: AES hardware accelerator (AES)</a></li> <li>– <a href="#">Section 21: Advanced-control timer (TIM1)</a></li> <li>– <a href="#">Section 22: General-purpose timers (TIM2/TIM3/TIM4)</a></li> <li>– <a href="#">Section 24: General-purpose timers (TIM14)</a></li> <li>– <a href="#">Section 25: General-purpose timers (TIM15/TIM16/TIM17)</a></li> <li>– <a href="#">Section 26: Low-power timer (LPTIM)</a></li> <li>– <a href="#">Section 32: Inter-integrated circuit interface (I2C)</a></li> <li>– <a href="#">Section 33: Universal synchronous receiver transmitter (USART)</a></li> <li>– <a href="#">Section 34: Low-power universal asynchronous receiver transmitter (LPUART)</a></li> <li>– <a href="#">Section 31: Tamper and backup registers (TAMP)</a></li> <li>– <a href="#">Section 35: Serial peripheral interface / integrated interchip sound (SPI/I2S)</a></li> <li>– <a href="#">Section 37: Universal serial bus full-speed host/device interface (USB)</a></li> <li>– <a href="#">Section 36: FD controller area network (FDCAN)</a></li> <li>– <a href="#">Section 41: Device electronic signature</a></li> </ul>

**Table 267. Document revision history (continued)**

Date	Revision	Changes
09-Dec-2024	6	<p>Added errata sheet in the list of reference documents as well as the mention that patents apply to the microcontrollers on document cover page.</p> <p>Updated DMAMUX register block size in <a href="#">Table 6: STM32G0x1 peripheral register boundary addresses</a>.</p> <p>Updated parity check in <a href="#">Section 2.3: Embedded SRAM</a>.</p> <p>Added caution note related to BOOT0 pin in <a href="#">Section 2.5: Boot configuration</a>.</p> <p>Updated <a href="#">Section 2.5.4: Empty check</a>.</p> <p><b>Section 3: Embedded flash memory (FLASH)</b></p> <p>Updated main memory block description in <a href="#">Section 3.3.1: FLASH organization</a>. Modified page numbers corresponding to Bank 2 in <a href="#">Table 10: Flash memory organization for 256 KB dual-bank devices</a> and <a href="#">Table 11: Flash memory organization for 512 KB devices</a>.</p> <p>Updated <a href="#">Section 3.3.3: FLASH error code correction (ECC)</a>.</p> <p>Updated conditions in which PGAERR and WRPERR errors can occur in <a href="#">Section : Programming errors</a>.</p> <p>Updated <a href="#">Section : Modifying user options</a>.</p> <p>Updated memory areas protected by RDP in <a href="#">Section 3.5.1: FLASH read protection (RDP)</a>.</p> <p>Updated PNB[9:0] bitfield description of <a href="#">FLASH control register (FLASH_CR)</a>.</p> <p>Updated SEC_SIZE[7:0] description of <a href="#">FLASH security register (FLASH_SECR)</a>.</p> <p><b>Section 4: Power control (PWR)</b></p> <p>Replaced “power voltage detector” by “programmable voltage detector” in PVDRT, PVDFT and PVDE bit descriptions of <a href="#">Power control register 2 (PWR_CR2)</a> and PVDO of <a href="#">Power status register 2 (PWR_SR2)</a>.</p> <p><b>Section 6: Clock recovery system (CRS)</b></p> <p>Updated <a href="#">CRS control register (CRS_CR)</a>.</p> <p><b>Section 5: Reset and clock control (RCC)</b></p> <p>Updated <a href="#">Section 5.1.2: System reset</a> to indicate the a system reset sets all registers to their reset value unless otherwise specified.</p> <p>Modified <a href="#">Section : External source (HSE bypass)</a>.</p> <p><b>Section 12: Nested vectored interrupt controller (NVIC)</b></p> <p>Updated <a href="#">Section 12.2: SysTick calibration value register</a>.</p> <p><b>Section 14: Cyclic redundancy check calculation unit (CRC)</b></p> <p>Updated <a href="#">Figure 30: CRC calculation unit block diagram</a>.</p> <p>Added note in <a href="#">Section : Polynomial programmability</a> to clarify what are even and odd polynomials.</p> <p>Added CRC register access granularity in <a href="#">Section 14.2: CRC main features</a> and <a href="#">Section 14.4: CRC registers</a>.</p>

Table 267. Document revision history (continued)

Date	Revision	Changes
09-Dec-2024	6 (continued)	<p><b>Section 15: Analog-to-digital converter (ADC)</b>  Removed ADC supply requirements from <a href="#">Section 15.2: ADC main features</a>  <a href="#">Table 71: ADC input/output pins</a>: removed “<math>V_{DDA} \geq V_{DD}</math>” from <math>V_{DDA}</math> description as well as the recommendation to connect <math>V_{SSA}</math> to <math>V_{SS}</math>.  Removed JOFFSETx and JOFFSETx_CH from <a href="#">Figure 31: ADC block diagram</a> since offset control is not available.  Updated software calibration procedure in <a href="#">Section 15.3.3: Calibration (ADCAL)</a>.  Updated note related to ADSTART clearing in <a href="#">Section 15.3.12: Starting conversions (ADSTART)</a>.  Updated formula to calculate the temperature in <a href="#">Section : Reading the temperature</a>.  Changed value of converted digital value to <math>V_{BAT}/3</math> and updated <a href="#">Figure 58: VBAT channel block diagram</a> in <a href="#">Section 15.10: Battery voltage monitoring</a>.</p> <p><b>Section 16: Digital-to-analog converter (DAC)</b>  Added VREF+ pin availability in <a href="#">Table 82: DAC features</a></p> <p><b>Section 20: AES hardware accelerator (AES)</b>  Corrected IVI[127:0] bitfield split among AES_IVRx registers (<a href="#">Table 107: CTR mode initialization vector definition</a>).  Corrected <a href="#">Table 109: Initialization of AES_IVRx registers in GCM mode</a> and <a href="#">Table 110: Initialization of AES_IVRx registers in CCM mode</a>.  Updated text relative to the CCF flag in section <a href="#">DMA operation in different operating modes</a>.</p> <p><b>Section 21: Advanced-control timer (TIM1)</b>  Updated <a href="#">Figure 124: Control circuit in normal mode, internal clock divided by 1</a>.  Added Note in <a href="#">Section 21.3.16: Using the break function</a>.  Updated OC1PE in <a href="#">Section 21.4.8: TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)</a>.</p> <p><b>Section 24: General-purpose timers (TIM14)</b>  Updated OC1PE in <a href="#">TIM14 capture/compare mode register 1 [alternate] (TIM14_CCMR1)</a>.</p> <p><b>Section 25: General-purpose timers (TIM15/TIM16/TIM17)</b>  Updated <a href="#">Figure 247: Control circuit in normal mode, internal clock divided by 1</a>.  Added Note in <a href="#">Section 25.4.13: Using the break function</a>.  Added <a href="#">Section 25.4.15: 6-step PWM generation</a>.  Suppressed CC2DE in <a href="#">TIM15 DMA/interrupt enable register (TIM15_DIER)</a>.  Updated <a href="#">TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)</a>.  Updated <a href="#">TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)</a>.</p>

Table 267. Document revision history (continued)

Date	Revision	Changes
09-Dec-2024	6 (continued)	<p><b>Section 26: Low-power timer (LPTIM)</b>  Updated <a href="#">Section 26.2: LPTIM main features</a>.  Added note to <a href="#">Section 26.4.7: Trigger multiplexer</a>.  Updated <a href="#">Section 26.4.4: LPTIM reset and clocks</a>.  Updated <a href="#">Section 26.4.15: Encoder mode</a>.  Updated <a href="#">LPTIM interrupt and status register (LPTIM_ISR)</a>, <a href="#">LPTIM interrupt clear register (LPTIM_ICR)</a>. <a href="#">LPTIM configuration register (LPTIM_CFGR)</a></p> <p><b>Section 30: Real-time clock (RTC)</b>  Updated <a href="#">Section 30.3.13: RTC smooth digital calibration</a>.  Updated <a href="#">RTC initialization control and status register (RTC_ICSR)</a> and <a href="#">RTC status clear register (RTC_SCR)</a>.</p> <p><b>Section 32: Inter-integrated circuit interface (I2C)</b>  Updated <a href="#">Figure 284: Block diagram</a> and <a href="#">Figure 296: Transfer bus diagrams for I2C target receiver (mandatory events only)</a>.</p> <p><b>Section 33: Universal synchronous receiver transmitter (USART)</b>  In the whole document renamed SCLK pin into CK, and replaced nCTS and nRTS by CTS and RT, respectively.  Removed mention that usart_wkup interrupt is not mandatory when wake-up event is detected from <a href="#">Section 33.5.21: USART low-power management</a>.  Updated ADD[7:0] bitfield descriptions in <a href="#">USART control register 2 (USART_CR2)</a>.</p> <p><b>Section 34: Low-power universal asynchronous receiver transmitter (LPUART)</b>  In the whole document renamed SCLK pin into CK, and replaced nCTS and nRTS by CTS and RT, respectively.  Updated <a href="#">Table 198: Error calculation for programmed baud rates at lpuart_ker_ck_pres = 32.768 kHz</a>.  Removed mention that lpuart_wkup interrupt is not mandatory when wake-up event is detected from <a href="#">Section 34.4.14: LPUART low-power management</a>.  Updated ADD[7:0] bitfield descriptions in <a href="#">LPUART control register 2 (LPUART_CR2)</a>.</p> <p><b>Section 35: Serial peripheral interface / integrated interchip sound (SPI/I2S)</b>  Updated <a href="#">Section : Communication using DMA (direct memory addressing)</a>.  Updated <a href="#">Table 207: Audio-frequency precision using 48 MHz clock derived from HSE</a> title.</p> <p><b>Section 36: FD controller area network (FDCAN)</b>  Updated <a href="#">Figure 392: CAN subsystem</a>. and <a href="#">Section 36.3.6: Message RAM</a>.</p>

Table 267. Document revision history (continued)

Date	Revision	Changes
09-Dec-2024	6 (continued)	<p><b>Section 37: Universal serial bus full-speed host/device interface (USB)</b>  Updated <i>Table 228: Bulk double-buffering memory buffers usage (Device mode)</i>, <i>Table 229: Bulk double-buffering memory buffers usage (Host mode)</i> and <i>Table 230: Isochronous memory buffers usage</i>  Updated <i>USB endpoint/channel n register (USB_CHEPnR)</i>.</p> <p><b>Section 38: USB Type-C®/USB Power Delivery interface (UCPD)</b>  Added table footnote added in <i>Table 249: Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)</i>.  Removed UCPD_CFGR3 register.  Removed CC1VCONNEN and CC2VCONNEN bits from <i>UCPD control register (UCPD_CR)</i>.</p>

# Index

## A

ADC_AWD1TR	390
ADC_AWD2CR	394
ADC_AWD2TR	390
ADC_AWD3CR	395
ADC_AWD3TR	393
ADC_CALFACT	395
ADC_CCR	396
ADC_CFGR1	384
ADC_CFGR2	387
ADC_CHSELR	390-391
ADC_CR	382
ADC_DR	394
ADC_IER	380
ADC_ISR	379
ADC_SMPR	389
AES_CR	503
AES_DINR	507
AES_DOUTR	507
AES_IVR0	509
AES_IVR1	510
AES_IVR2	510
AES_IVR3	510
AES_KEYR0	508
AES_KEYR1	508
AES_KEYR2	509
AES_KEYR3	509
AES_KEYR4	511
AES_KEYR5	511
AES_KEYR6	511
AES_KEYR7	512
AES_SR	506
AES_SUSPxR	512

## C

CEC_CFGR	1346
CEC_CR	1345
CEC_IER	1350
CEC_ISR	1348
CEC_RXDR	1348
CEC_TXDR	1348
COMP1_CSR	446
COMP2_CSR	448
COMP3_CSR	450
CRC_CR	338
CRC_DR	337
CRC_IDR	337

CRC_INIT	339
CRC_POL	339
CRS_CFGR	223
CRS_CR	222
CRS_ICR	226
CRS_ISR	224

## D

DAC_CCR	431
DAC_CR	420
DAC_DHR12L1	424
DAC_DHR12L2	426
DAC_DHR12LD	427
DAC_DHR12R1	424
DAC_DHR12R2	425
DAC_DHR12RD	427
DAC_DHR8R1	425
DAC_DHR8R2	426
DAC_DHR8RD	428
DAC_DOR1	428
DAC_DOR2	429
DAC_MCR	431
DAC_SHHR	433
DAC_SHRR	434
DAC_SHSR1	433
DAC_SHSR2	433
DAC_SR	429
DAC_SWTRGR	423
DBG_APB_FZ1	1363
DBG_APB_FZ2	1365
DBG_CR	1362
DBG_IDCODE	1361
DMA_CCRx	289
DMA_CMARx	293
DMA_CNDTRx	291
DMA_CPARx	292
DMA_IFCR	287
DMA_ISR	285
DMAMUX_CFR	307
DMAMUX_CSR	307
DMAMUX_CxCR	306
DMAMUX_RGCFR	309
DMAMUX_RGSR	309
DMAMUX_RGxCR	308

## E

EXTI_EMR1	330
-----------	-----

## Index

EXTI_EMR2	331	FDCAN_TXFQS	1238
EXTI_EXTICRx	328	FDCAN_XIDAM	1234
EXTI_FPR1	325	FLASH_ACR	94
EXTI_FPR2	327	FLASH_CR	98
EXTI_FTSR1	323	FLASH_ECCR	100
EXTI_FTSR2	326	FLASH_ECCR2	101
EXTI_IMR1	329	FLASH_KEYR	95
EXTI_IMR2	330	FLASH_OPTKEYR	95
EXTI_RPR1	324	FLASH_OPTR	102
EXTI_RPR2	327	FLASH_PCROP1AER	105
EXTI_RTSR1	322	FLASH_PCROP1ASR	104
EXTI_RTSR2	325	FLASH_PCROP1BER	107
EXTI_SWIER1	323	FLASH_PCROP1BSR	107
EXTI_SWIER2	326	FLASH_PCROP2AER	108
		FLASH_PCROP2ASR	108
		FLASH_PCROP2BER	110
		FLASH_PCROP2BSR	110
		FLASH_SECR	111
		FLASH_SR	96
		FLASH_WRP1AR	105
		FLASH_WRP1BR	106
		FLASH_WRP2AR	109
		FLASH_WRP2BR	109
<b>F</b>		<b>G</b>	
FDCAN_CCCR	1218	GPIOx_AFRH	243
FDCAN_CKDIV	1243	GPIOx_AFRL	242
FDCAN_CREL	1215	GPIOx_BRR	243
FDCAN_DBTP	1215	GPIOx_BSRR	241
FDCAN_ECR	1223	GPIOx_IDR	240
FDCAN_ENDN	1215	GPIOx_LCKR	241
FDCAN_HPMS	1234	GPIOx_MODER	238
FDCAN_IE	1229	GPIOx_ODR	241
FDCAN_ILE	1232	GPIOx_OSPEEDR	239
FDCAN_ILS	1231	GPIOx_OTYPER	239
FDCAN_IR	1226	GPIOx_PUPDR	240
FDCAN_NBTP	1219		
FDCAN_PSR	1224	<b>I</b>	
FDCAN_RWD	1217	I2C_CR1	969
FDCAN_RXF0A	1236	I2C_CR2	972
FDCAN_RXF0S	1235	I2C_ICR	980
FDCAN_RXF1A	1237	I2C_ISR	978
FDCAN_RXF1S	1236	I2C_OAR1	974
FDCAN_RXGFC	1232	I2C_OAR2	975
FDCAN_TDCR	1226	I2C_PECR	981
FDCAN_TEST	1216	I2C_RXDR	981
FDCAN_TOCC	1222	I2C_TIMEOUTR	977
FDCAN_TOCV	1223	I2C_TIMINGR	976
FDCAN_TSCC	1221	I2C_TXDR	982
FDCAN_TSCV	1221	IWDG_KR	853
FDCAN_TXBAR	1239	IWDG_PR	854
FDCAN_TXBC	1237		
FDCAN_TXBCF	1241		
FDCAN_TXBCIE	1242		
FDCAN_TXBCR	1240		
FDCAN_TXBRP	1238		
FDCAN_TXBTIE	1241		
FDCAN_TXBTO	1240		
FDCAN_TXEFA	1243		
FDCAN_TXEFS	1242		



IWDG_RLR	855	RCC_APBENR1	194
IWDG_SR	856	RCC_APBENR2	198
IWDG_WINR	857	RCC_APBSTR1	188
		RCC_APBSTR2	191
<b>L</b>		RCC_APBSMENR1	201
LPTIM_ARR	845	RCC_APBSMENR2	205
LPTIM_CFGR	840	RCC_BDCR	209
LPTIM_CFGR2	846	RCC_CCIPR	206
LPTIM_CMP	845	RCC_CCIPR2	208
LPTIM_CNT	846	RCC_CFGR	176
LPTIM_CR	843	RCC_CICR	185
LPTIM_ICR	839	RCC_CIER	182
LPTIM_IER	839	RCC_CIFR	183
LPTIM_ISR	838	RCC_CR	174
LPUART_BRR	1111	RCC_CRRCR	182
LPUART_CR1	1100, 1103	RCC_CSR	211
LPUART_CR2	1106	RCC_ICSCR	176
LPUART_CR3	1108	RCC_IOPENR	192
LPUART_ICR	1119	RCC_IOPRSTR	186
LPUART_ISR	1112, 1116	RCC_IOPSMENR	199
LPUART_PRESC	1121	RCC_PLLCFGR	179
LPUART_RDR	1120	RNG_CR	463
LPUART_RQR	1111	RNG_DR	465
LPUART_TDR	1120	RNG_SR	464
		RTC_ALRMAR	895
<b>P</b>		RTC_ALRMASR	896
PWR_CR1	137	RTC_ALRMBR	897
PWR_CR2	138	RTC_ALRMBSSR	898
PWR_CR3	140	RTC_CALR	891
PWR_CR4	141	RTC_CR	887
PWR_PDCRA	146	RTC_DR	883
PWR_PDCRB	147	RTC_ICSR	884
PWR_PDCRC	148	RTC_MISR	899
PWR_PDCRD	149	RTC_PRER	886
PWR_PDCRE	150	RTC_SCR	900
PWR_PDCRF	151	RTC_SHIFTR	892
PWR_PUCRA	146	RTC_SR	898
PWR_PUCRB	147	RTC_SSR	884
PWR_PUCRC	148	RTC_TR	882
PWR_PUCRD	149	RTC_TSDR	893
PWR_PUCRE	150	RTC_TSSSR	894
PWR_PUCRF	151	RTC_TSTR	893
PWR_SCR	145	RTC_WPR	890
PWR_SR1	142	RTC_WUTR	887
PWR_SR2	143		
		<b>S</b>	
<b>R</b>		SPIx_CR1	1170
RCC_AHBENR	193	SPIx_CR2	1172
RCC_AHBSTR	187	SPIx_CRCPR	1176
RCC_AHBSMENR	200	SPIx_DR	1176
		SPIx_I2SCFGR	1177

## Index

---

SPIx_I2SPR	1179	TIM1_BDTR	600
SPIx_RXCR	1176	TIM1_CCER	594
SPIx_SR	1174	TIM1_CCMR1	587-588
SPIx_TXCR	1177	TIM1_CCMR2	591-592
SYSCFG_CFGR1	246	TIM1_CCMR3	606
SYSCFG_CFGR2	249	TIM1_CCR1	598
SYSCFG_ITLINE0	251	TIM1_CCR2	599
SYSCFG_ITLINE1	252	TIM1_CCR3	599
SYSCFG_ITLINE10	256	TIM1_CCR4	600
SYSCFG_ITLINE11	256	TIM1_CCR5	607
SYSCFG_ITLINE12	257	TIM1_CCR6	608
SYSCFG_ITLINE13	257	TIM1_CNT	597
SYSCFG_ITLINE14	258	TIM1_CR1	576
SYSCFG_ITLINE15	258	TIM1_CR2	577
SYSCFG_ITLINE16	258	TIM1_DCR	604
SYSCFG_ITLINE17	259	TIM1_DIER	582
SYSCFG_ITLINE18	259	TIM1_DMAR	605
SYSCFG_ITLINE19	260	TIM1_EGR	586
SYSCFG_ITLINE2	252	TIM1_OR1	606
SYSCFG_ITLINE20	260	TIM1_PSC	597
SYSCFG_ITLINE21	260	TIM1_RCR	598
SYSCFG_ITLINE22	261	TIM1_SMCR	580
SYSCFG_ITLINE23	261	TIM1_SR	584
SYSCFG_ITLINE24	262	TIM1_TISEL	612
SYSCFG_ITLINE25	262	TIM14_ARR	730
SYSCFG_ITLINE26	262	TIM14_CCER	728
SYSCFG_ITLINE27	263	TIM14_CCMR1	725-726
SYSCFG_ITLINE28	263	TIM14_CCR1	730
SYSCFG_ITLINE29	264	TIM14_CNT	729
SYSCFG_ITLINE3	252	TIM14_CR1	722
SYSCFG_ITLINE30	264	TIM14_DIER	723
SYSCFG_ITLINE31	264	TIM14_EGR	724
SYSCFG_ITLINE4	253	TIM14_PSC	730
SYSCFG_ITLINE5	254	TIM14_SR	723
SYSCFG_ITLINE6	254	TIM14_TISEL	731
SYSCFG_ITLINE7	254	TIM15_AF1	794
SYSCFG_ITLINE8	255	TIM15_ARR	788
SYSCFG_ITLINE9	255	TIM15_BDTR	790
		TIM15_CCER	785
		TIM15_CCMR1	781-782
		TIM15_CCR1	789
		TIM15_CCR2	790
		TIM15_CNT	788
		TIM15_CR1	773
		TIM15_CR2	774
		TIM15_DCR	793
		TIM15_DIER	777
		TIM15_DMAR	793
		TIM15_EGR	780
		TIM15_PSC	788
		TIM15_RCR	789
		TIM15_SMCR	776
<b>T</b>			
TAMP_BKPxR	917		
TAMP_CR1	910		
TAMP_CR2	911		
TAMP_FLTCR	912		
TAMP_IER	913		
TAMP_MISR	915		
TAMP_SCR	916		
TAMP_SR	914		
TIM1_AF1	609		
TIM1_AF2	611		
TIM1_ARR	597		

TIM15_SR	778	USART_BRR	1051
TIM15_TISEL	795	USART_CR1	1036, 1039
TIM16_AF1	816	USART_CR2	1043
TIM16_TISEL	817	USART_CR3	1047
TIM17_AF1	818	USART_GTPR	1051
TIM17_TISEL	819	USART_ICR	1065
TIM2_AF1	686	USART_ISR	1054, 1060
TIM2_OR1	684	USART_PRESC	1068
TIM2_TISEL	688	USART_RDR	1067
TIM3_AF1	687	USART_RQR	1053
TIM3_OR1	685	USART_RTOR	1052
TIM3_TISEL	688	USART_TDR	1067
TIM4_AF1	687	USB_BCDR	1278
TIM4_OR1	686	USB_CHEP_RXTXBD_n	1293-1294
TIM4_TISEL	689	USB_CHEP_TXRXBD_n	1291
TIMx_ARR	681, 705, 810	USB_CHEPnR	1280
TIMx_BDTR	812	USB_CNTR	1270
TIMx_CCER	678, 807	USB_DADDR	1277
TIMx_CCMR1	672, 674, 804-805	USB_FNR	1276
TIMx_CCMR2	676-677	USB_ISTR	1273
TIMx_CCR1	681, 811	USB_LPMCSR	1278
TIMx_CCR2	682		
TIMx_CCR3	682		
TIMx_CCR4	683		
TIMx_CNT	679-680, 704, 809		
TIMx_CR1	662, 701, 799		
TIMx_CR2	663, 703, 800		
TIMx_DCR	684, 815		
TIMx_DIER	668, 703, 801		
TIMx_DMAR	684, 815		
TIMx_EGR	671, 704, 803		
TIMx_PSC	680, 705, 810		
TIMx_RCR	811		
TIMx_SMCR	665		
TIMx_SR	669, 704, 802		

**V**

VREFBUF_CCR	438
VREFBUF_CSR	438

**W**

WWDG_CFR	863
WWDG_CR	862
WWDG_SR	864

**U**

UCPD_CFGR1	1317
UCPD_CFGR2	1319
UCPD_CR	1320
UCPD_ICR	1327
UCPD_IMR	1322
UCPD_RX_ORDEXTR1	1331
UCPD_RX_ORDEXTR2	1332
UCPD_RX_ORDSETR	1329
UCPD_RX_PAYSZR	1330
UCPD_RXDR	1331
UCPD_SR	1324
UCPD_TX_ORDSETR	1328
UCPD_TX_PAYSZR	1328
UCPD_TXDR	1329

**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.

© 2024 STMicroelectronics – All rights reserved